DON XML Working Group

DON XML Developer's Guide Version 1.1

This Version:

DON XML Developer's Guide Version 1.1 – 1 May 2002

Latest Version:

DON XML Developer's Guide Version 1.1 – 1 May 2002

Previous Version:

Initial DON XML Developer's Guide – 29 October 2001

Author:

Brian Hopkins (xosys@sbcglobal.net)

About This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the NavyXML Quickplace[i]. Additional DON XML policy and guidance can also be found at the NavyXML Quickplace.

A version number has been introduced in the title of this document. The initial release of the document on 29 October 2001 represented version 1.0. This update is version 1.1. It represents the consensus of the DON XML WG as guidance for the development of XML components with the department.

This document is an early deliverable of the overall DON XML strategy for employing XML within the department. It provides general development guidance for the many XML initiatives currently taking place within the DON while the DON XML Work Group (DON XML WG) is in the process of developing a long-term strategy for aligning XML implementations with the business needs of the department. It is intended to be a living document that will be updated frequently.

This version of the guidance is primarily written to assist developers in creating schemas that describe XML payloads of information. It should be noted that payloads represent only one component required for secure, reliable information exchange. Other components include a specification for reliable messaging (including authentication, encryption, queuing, and error handling), business service registry and repository functions, and transport protocols. Emerging technologies and specifications are, or will shortly, provide XML-based solutions to many of these needs. The DON XML WG is developing an XML Primer that will describe each of these components and bring together the overall strategy for capitalizing on XML as a tool for enterprise interoperability.

Paragraphs of this document are broken into three parts.

- ♦ "Guidance" provides a concise summary of requirements and recommendations.
- ♦ "Explanation" provides a brief explanation of the reasoning behind the guidance provided.
- ♦ "Example" provides one or more non-normative examples pertaining to the guidance.

The bulk of this document is contained in appendices that are provided as non-normative supplementary information. The appendices should be considered to have a "draft" status, and do not represent the consensus of the DON XML Working Group (WG).

This document is primarily intended for developers already familiar with XML; however, it has a comprehensive glossary that provides good starting points for XML beginners. Some of this document focuses on XML Schemas as a tool for interoperability. To get the maximum benefit, it is suggested that you take the time to become familiar with the XML Schema language. An excellent tutorial with labs is available at http://www.xfront.com/.

The DON XML WG encourages developers to try the techniques recommended here and provide feedback via the editor. Lessons learned and best practices will be collected and used to update and expand the guide periodically.

Table of Contents

1. References

(a) DON CIO Interim Policy on the Use of Extensible Markup Language For Data Exchange dtd 06 Sept 2001

(b) DON XML Vision dtd 15 March 2002.

(c) SECNAVINST 5000.36, Data Management and Interoperability

2. Introduction

In August 2001 DON CIO established a DON XML Work Group (DON XML WG) to provide the leadership and guidance to maximize the value and effectiveness of emerging XML component technologies implemented across the DON Enterprise. At its first meeting in August 2001, the DON XML WG agreed to produce a DON XML Developer's Guide as a deliverable. This document serves as a reference guide for making existing applications "*XML-enabled",* and for developing future capabilities that will leverage XML to the maximum extent possible.

Service initiatives such as Task Force Web[ii] (TFWeb) are implementing XML-enabled applications very quickly. This document will assist DON activities in developing XML implementations in the short term, while lessons learned are collected.

On 6 September 2001 the Department of the Navy Chief Information Officer signed out reference (a), an Interim XML Policy Statement on the use of XML within the department. Copies of this policy are available on the NavyXML QuickPlace.

On 15 March, the DON CIO released reference (b), a vision statement for XML:

"In order to achieve maritime superiority, the Department of the Navy will fully exploit Extensible Markup Language technology as a key interoperability tool for next generation DON knowledge superiority and its developing network centric information infrastructure".

Subsequently, the DON XML WG divided into 5 action teams. The purpose of Action Team 2 (AT 2) is:

"To support the Department of the Navy's (DONs) vision to fully exploit Extensible Markup Language (XML) as an enabling technology to achieve interoperability in support of maritime information superiority by developing policy, guidance and procedures to establish a standard framework for ***organization specific XML implementation."***

This Guidance is an early deliverable of AT 2 and will continue to be updated and expanded by it during the course of the DON XML WG's existence.

3. Terminology and Conventions

The terms "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**MAY**", and "**OPTIONAL**" are used throughout this document, and should be interpreted in accordance with the Harvard University Network Group "Request for Comments" #2119 Best Current Practices" #14 (RFC 2119i)[iii]

The term XML is used throughout this document to describe a large range of specifications and technologies associated with XML markup.

It is critical that activities developing XML-enabled applications have a firm understanding of basic XML terminology. Appendix G provides a list of applicable acronyms and terms.

Many schema languages have been created for expressing XML validation rules; however, throughout this document the term 'schema' with a small 's' is used to generically refer to all XML Validation languages (to include DTDs), while the term XML Schema or just Schema (capital 'S') refers specifically to schemas authored in accordance with the W3C XML Schema recommendation.

4. Implementation Requirements

This document defines a standard for using XML within the DON. It provides recommendations and best practices for the creation of XML schema and components for "XML-enabling" applications.

DON CIO understands that short timeframe XML implementations (such as TFWeb), or pre-existing schema that do not follow this guidance cannot be changed immediately. Activities **SHOULD** read this document and develop a migration plan to evolve their current XML implementations; additionally, the DON XML WG encourages submission of feedback as lessons learned are collected.

4.1. Requirements Level

The RFC 2119 terms defined above should be interpreted in the context of this document's requirements level, which is that of guidance.

4.2. Conformance

Enforcing conformance to the requirements of this document is, at present, left to the discretion of the program manager. As this document matures, the DON CIO **MAY** elevate some or all of the guidance to a higher requirements level.

4.3. Conflict resolution

In the event of a conflict between this document and other Navy standards, this document **SHOULD** have precedence for matters pertaining to XML only.

4.4. Applicability

This guidance applies to all activities in the DON that are implementing applications that use XML for the exchange of information with other applications via public interfaces. This version of the developers guide contains guidance of a general nature that is applicable to both document-centric and data-centric information exchanges. It also contains specific guidance for data-centric exchanges necessary for enterprise interoperability. Specific guidance for document-centric applications will be forthcoming in the next version.

These recommendations are not intended to restrict the use of XML internal to systems; the DON XML WG recommends that applications separate internal XML grammars processed by application code from that used for external communications. This decoupling of internally processed XML with that which is communicated externally insulates application code from XML vocabulary evolution and allows such loosely coupled applications to stay current with the latest schemas and components promulgated by communities of interest and Voluntary Consensus Standards.

5. DoD XML Registry

Guidance

Reference (a) **REQUIRES** all DON developers to reuse Voluntary Consensus Standard vocabularies if applicable, or reuse existing tags in the DoD XML Registry, if sufficient, or before developing their own.

Reference (a) **REQUIRES** activities to register developed XML Components with the DOD XML Registry.

Emerging DoD XML policy is *expected to require* registration of Voluntary Consensus Standard components; therefore activities **SHOULD** include these components in their registration packages.

Developers **MUST** familiarize themselves with DoD XML Registry site and the associated DoD Namespaces[1]. Each activity submitting a registration package to the registry is **REQUIRED** to do so to a specific DOD Namespace via the Namespace Manager. In the case where an application's data crosses DoD Namespace boundaries, activities **SHOULD** request the DoD Namespace Manager to provide guidance.

Explanation

While this guidance provides many recommendations and examples of how to create more interoperable XML, the single biggest factors affecting interoperability

[1] A COE Namespace and an XML Namespace are not the same thing. It is important to understand the difference. The difference is explained in the Appendix G – Draft Glossary under COE Namespace.

are visibility and reuse. A draft DoD policy establishes the Defense Information Systems Agency (DISA) as the lead for the single DoD point of entry for XML registry and repository functions.. The intent of the DOD Registry is to provide visibility into XML components that are being used throughout the DoD.

The DON XML WG is working with DoD representatives to develop specific guidance for developers as to which DoD Namespace they should register with. Until this is promulgated, activities should study the Namespace descriptions on the registry site and contact the Namespace manager for what appears to be the most appropriate place for registration. If unable to locate an appropriate Namespace, register with the '*To Be Determined*' (TBD) Namespace.

Example

An example of a DoD Registration package from the DoD XML Registry is available for download from the NavyXML Quickplace library.

6. Recommended XML Specifications

Guidance

Standards promulgated by nationally or internationally accredited standards bodies (such as ISO, IEEE, ANSI, OASIS, UN/CEFACT, IETF, etc.) **MUST** be adhered to when developing applications within the domain that the standard addresses. The only exception to this rule is when a standard produced by one of these bodies competes with a similar product of the W3C. In this case, only, the W3C has precedence.

In general, production applications **SHOULD** only use software that implements W3C Final Recommendations and final specifications of the accredited standards bodies referenced in the above paragraph. Applications using software that implements W3C technical reports at other stages of the development process or other draft products of Voluntary Consensus Standards bodies **MUST** do so with the following restrictions:

- ♦ Production Applications:
  - ➢ Prior to creating, incorporating or using software that implements non-W3C specifications, activities **MUST**:
    - ▪ Ensure that no competing W3C endorsed final recommendation exists or is being developed (and is at least at the Second Work Draft level). Future revisions of this document will provide more specific guidance.
    - ▪ Ensure that the specification is a product of an accredited standards body (ISO, IEEE, ANSI, UN/CEFACT, IETF) or a credible Voluntary

    Consensus Standards body such as OASIS, the OMG, OAG, UDDI, RossettaNet, or BizTalk . The decision of what is considered credible organizations is, for the time being, up to the government program manager.

  - Activities **MAY** choose to implement W3C technical reports with a *Proposed Recommendation* status provided they are committed to immediately update software should any changes be made when the report reaches final status.

- Pilot Applications:

  - Activities developing pilot applications (as a precursor to production) **MAY** also implement software that conforms to W3C technical reports with a *Candidate Recommendation* status.

- (Advanced Concept) Demonstrations:

  - Activities developing demonstration applications (as a proof of concept) **MAY** also implement software that conforms to W3C technical reports with a *Working Draft* or *Note* status or another accredited standards body or Voluntary Consensus Standards body's draft specifications.

  - Exception:

    - Activities **MAY** implement software that conforms to the SOAP 1.1 *W3C Note*, but **MUST** then be ready and committed to update software to the SOAP 1.2 specification when it reaches *Final Recommendation* status.

    - Activities MAY implement the SAX 1.0 and 2.0.

All software and software components (XML parsers, generators, validators, enabled applications, servers, databases, operating systems), and other software acquired or used by DON activities **SHALL** be fully compliant with all W3C XML technical reports holding final recommendation status and with final specifications produced by accredited standards bodies.

Proprietary extensions to W3C Technical Reports or other specifications by accredited standards bodies :

- **MUST NOT** be employed in any software or XML document (instance, schema, style sheet) that will be shared publicly with activities outside a local development environment.

- **SHOULD** only be employed locally (within a homogeneous development environment) after careful evaluation of possible impacts on cross-platform interoperability, and dependency on software from a single vendor.

- Government program managers **MUST** have the final say in the decision to employ such extensions, even when doing so inside a single system's boundaries or within a homogeneous development environment.

Explanation

In order to promote interoperability on the widest possible scale, Internationally accredited standards bodies must have precedence over other organization's technical products with the exception of the W3C. The W3C is a vendor consortium, not an accredited standards body, however, its products have such a strong influence over commercial software implementations that its work must take precedence over even accredited standards bodies for matters relating to the World Wide Web (including XML even though XML is restricted to the WWW.)

OASIS is not currently and an accredited standards organization, it is officially a Voluntary Consensus Standards body, however OASIS has signed a memorandum of understanding with ISO and IEEE, and has been given official liaison standing with these organizations. Consequently, the DON considers OASIS to same status as accredited standards bodies.

EbXML is neither an accredited standards body nor a Voluntary Consensus Standards body. EbXML was an 18-month project sponsored by UN/CEFACT and OASIS. After completion of the project in May 2001, the work of ebXML is being carried forward by UN/CEFACT and OASIS jointly.

The W3C Technical Reports page has a complete list of W3C reports in all stages of development. The following table provides a list of XML specifications or standards that are not W3C recommendations (yet). Two categories are provided. The "Recommended" column represents widely adopted standards that are believed to be mature and uniformly supported by software implementations. The "Maturing" column represents other standards that the DON XML WG believes to be sufficiently mature; however, they may not be uniformly supported in existing software implementations, so caution is advised. Future versions of this document will add additional specifications from other standards bodies and efforts such as ebXML, OASIS, UN/CEFACT, etc.

Recommended	***Maturing***
SAX[2] 1.0 and 2.0	SOAP 1.1 (W3C Note)

SOAP 1.1 has been adopted by various commercial and DON activities such as ebXML and TFWeb; therefore members of the DON XML WG have evaluated the

[2] SAX is not a specification developed by a standards body or the W3C. It is an open source project maintained by a community of developers. SAX parsers have been written for several languages, but the only platform independent version is the Java API. A parser that is SAX compliant must implement an equivalent to the Java API, which is provided at the SAX homepage.

specification and believe that it is sufficiently stable and mature to support production implementation. SOAP 1.1 exists as a W3C Note; however SOAP 1.2 is being pursued by the W3C XML Protocol Working Group[iv]. When it becomes a Final Recommendation, activities with SOAP 1.1 implementation must have planned for and be ready to migrate to SOAP 1.2.

The Simple API for XML, SAX, is a widely adopted specification that is the product of a software developer consortium. It is mature, stable, widely implemented in XML parsers and well managed in the open source environment.

Application vendors often provide proprietary extensions to adopted standards. These extensions may simplify the job of software developers, but they also make developed systems dependent on software from a single vendor, and often they also restrict the software to being run on a single vendor's operating system or hardware. The decision to employ these extensions in any DON application must be made by the government program manager after careful consideration of the interoperability impacts.

Example

An example of a conflict between OASIS standards and the W3C exists with respect to XML schema languages. The W3C promulgated XML Schema language and the OASIS promulgated RELAX-NG language. While the DON XML WG recognizes that competing standards such as RELAX-NG may have technical merit when compared with W3C products, the WG also realized the value in standards conformance, and as such has designated the W3C as the authoritative source for specifications related to XML and the World Wide Web.

To further illustrate the guidance regarding use of proprietary extensions to W3C Technical Reports, two examples are provided:

- Example 1: An activity developing an XSL stylesheet is using the XALAN XSL processor. Developers discover that the XALAN software has implemented an extension to XSLT that allows generation of multiple output HTML documents from a single stylesheet. This is convenient since the project requires multiple outputs. The lead project manager consults with the government program manager; the program manager agrees to allow the use of this proprietary extension provided a stylesheet without the extension is also delivered.

- Example 2: An activity is developing a Visual Basic application for deployment in a Windows 2000 environment. In that application, the MSXML DOM API is used to manipulate XML. Microsoft has added many convenient extensions to the W3C DOM recommendation that the developers want to use. Since the programming environment is restricted to the Microsoft environment (Windows and Visual Basic), the government program manager agrees to allow the use of the MSXML DOM.

They key difference between these examples is software code portability. In the first example, the stylesheet delivered should be able to run in any environment (operating system, language and XSL processor); therefore a strictly XSLT conformant deliverable was required. In the second example, code portability was not an issue since the project was restricted to the Microsoft environment already due to the choice of programming language and operating system.

7. XML Conventions

7.1. XML Components

7.1.1. Standardized Case Convention

Guidance

DON developers **SHALL** adopt the camel case convention, as defined by the ebXML Technical Architecture, when creating XML component names.

- XML Elements and XML Schema data types use upper camel case: The first letter in the name is upper case, as is the letter beginning each subsequent word.
- XML Attributes use lower camel case: Like upper camel case, except the first letter of the first word is lower case.

Explanation

Voluntary Consensus Standards bodies and other XML organizations such as OASIS, RosettaNet, Biztalk and ebXML (see Internet references in Appendix C) have all adopted the camel case convention for XML component naming, with ebXML differentiating between upper and lower camel case.

Example

```
 <?xml version="1.0" encoding="UTF-8" ?>
<!--
 Example of an upper camel case element and lower camel case
  attribute
 -->
 <UpperCamelCaseElement
                 lowerCamelCaseAttribute="foo" />
```

7.1.2. Usage of Acronyms and Abbreviations

Guidance

DON developers **SHOULD** follow the ebXML guidance for usage of acronyms or abbreviations in XML component names with the following caveats:

- Acronyms and abbreviations **SHOULD** generally be avoided in XML element and attribute names.
- For XML Schema data types, abbreviations **MUST** be avoided while acronyms **MAY** be used consistent with the rest of this guidance.
- When acronyms are used they **MUST** be in upper case. Abbreviations **SHOULD** be treated as words and expressed in upper camel case.
- While commonly used acronyms and abbreviations **MAY** be used in element and attribute names; the decision to use an acronym or abbreviation **SHALL** be made by program managers rather than by application developers . The decision to use an acronym or abbreviation **MUST** be based on the belief that its use will promote common understanding of the information both inside a community of interest as well as across multiple communities of interest. When an acronym or abbreviation does not come from a credible, identifiable source or when it introduces a margin for interpretation error, it **MUST NOT** be used.
- Acronyms and abbreviations used in component names **MUST** be spelled out in the component definition that is required to be included via schema annotations (as XML comments or inside XML Schema annotation <xsd:documentation> elements) (see Section 7.2.3.2). References to authoritative sources from which the acronyms or abbreviations are taken **SHOULD** also be included in schema documentation.

Explanation

XML documents that rely heavily on terse abbreviated component names are difficult to understand and subject to misinterpretation. The general consensus among the major XML standards development consortia is that abbreviations should be avoided and acronyms used sparingly. Within the DON, business language is heavily laden with both acronyms and abbreviations and it is often difficult to distinguish between an acronym and an abbreviation (e.g., CONOPS). After significant deliberation, the DON XML WG adopted the position that acronyms and abbreviations for use in element and attribute names are acceptable where they make sense, but should in general be avoided. While allowing usage, the working group strongly recommends that the decision for usage be based on a management decision that such usage will actually promote understanding. The DON XML WG is addressing the issue of authoritative abbreviation sources as part of the reference (c) Functional Data

Manager responsibilities. For the purpose of this document, authoritative source determination for abbreviations is left to program manager's discretion.

Example:

This is an example of providing an element definition in a DTD. Note that the acronym DoD is spelled out in the definition.

```
 <!-- DODActivityAddressCode
Definition: A 6-digit code used to uniquely identify
organizations within the Department of Defense (DoD)
-->
<!ELEMENT DODActivityAddressCode (#PCData)>
```

7.1.3. XML Component Selection and Creation

Guidance

Each DON organization **MUST** select, use, and adhere to appropriate Voluntary Consensus Standards (VCSs), consistent with PL 104-113[v] and OMB A-119[vi] (i.e., use suitable existing VCSs in lieu of developing new DoD or DON XML components).

DON organizations **SHALL** only develop DON XML components when they are needed to support DON technical and programmatic needs and when

(1) Suitable VCSs do not exist;

(2) Existing VCSs do not suffice or are not appropriate for the intended application; or

(3) A new VCS cannot be readily developed through a standards development organization (SDO).

(4) Suitable DoD components do not exist;

(5) Existing DoD components do not suffice or are not appropriate for the intended application; and

(6) New DoD components cannot be developed through the appropriate DoD standards process.

Reference (a) requires that existing DoD XML components be used if suitable. Therefore, the DoD XML registry **MUST** be searched for existing suitable components prior to creation of new components. There are three possible results for this search. Components may be fully or partially suitable, or no component may be found.

- A component is suitable if:

  - It satisfies the element domain requirements,
  - It is in upper/lower camel case depending on whether it is an element, attribute or type,
  - Is either named after a “business term”, or conforms to ISO 11179 conventions and
  - Abbreviations and acronyms are spelled out in the component definition.

If the component is suitable, it **MUST** be used. Use of that component **MUST** be registered within the DoD XML Registry when/if the registry supports it.

When a DoD component exists but is not suitable, the following procedure can be used to derive a suitable component while maintaining relationships to existing DoD components.

- Create an XML component using the following steps:
  - A “dictionary entry” using the ISO 11179 rules as modified by ebXML and the eBTWG (see Appendix A) **SHOULD** be created for each class or entity and each attribute of the classes/entities from a logical model of the information exchange requirement.
  - [XML Schema only] An XML Schema Type **SHOULD** be derived from an ISO 11179-compliant name (see 7.1.3.2 Creating XML Component Names from ISO 11179 Data Elements).[3] The type **SHOULD** be documented with metadata from the DoD registry entry upon which this suitable component is derived. Metadata **SHOULD** include items such as the definition, URL to the item, and registry identifier. Any domain restrictions **SHOULD** be applied to the type rather than the element. Additionally, mappings to authoritative DON or DoD data models or data element definitions (such as the DDDS) **MAY** be documented in the element's definition (see section 7.2.3.2, Capturing XML Component Definitions).
  - Element Creation
    - XML Schemas: Create an XML Element that is named according to a business term (see 7.1.3.1 Creating XML Element Names from Business Terms).The element **SHOULD** reference the ISO 11179-derived type created above. In the case where no suitable business term exists use the ISO 11179-derived type name (see 7.1.3.2 Creating XML Component Names from ISO 11179 Data Elements) .

[3] When used as XML component names, ISO 11179 element names **SHALL** be converted to camel case by removing the periods and spaces and adjusting the capitalization.

Create an XML Element using the DoD element name and declare it in the substitution group of the element created above.

- DTDs: Create elements that are named after business terms or ISO 11179-compliant names. Document the DoD Registry element name and the ISO 11179 name (if a business term is used) in the DTD as an XML comment.

➢ Attribute Creation: An ISO 11179-compliant names **SHOULD** be created for items that are represented as attributes (see 7.1.3.2 Creating XML Component Names from ISO 11179 Data Elements). XML Attributes **SHOULD** be selected based on the guidance of Section 7.4 – Attributes Vs. Elements, *not on their correspondence with data model attributes*.

♦ Register the new element and its relationship to the existing DoD element in the appropriate namespace of the DoD XML Registry.

♦ If no component is found, XML component names **SHOULD** be created following the rules defined above for unsuitable components, except that there will be no reference to an existing DoD Registry element.

Explanation

The Interim DON XML policy [reference (a)] requires the reuse of XML elements registered in the DoD XML Registry if those tags are found suitable. The intent of this guidance is to provide clarification as to what suitability means, and to reinforce the mandate that the registry be searched as a starting point for suitability determination.

In the case where an element has been identified as a candidate for reuse but fails suitability criteria, the above guidance provides a solution for creation of a suitable element while maintaining a semantic relationship to the initially discovered candidate.

For creation of XML elements when no suitable element exists in the DoD registry, the DON XML WG recommends the ebXML-modified ISO 11179 data element naming convention as a solid basis for XML component creation; however more commonly understood business terms can be used as element names, with the ISO 11179 structure preserved by XML Schema data types.

In summary, an ISO 11179 compliant data element name consists of three parts:

♦ An "Object Class" term, which describes the kind of thing being referred to. This Object Class may consist of one or more words, some of which may be context terms.

For example, the ISO 11179 name '***Acoustic Signal. Frequency. Measure***' has the "object class" '***Acoustic Signal***'.

- A “Property Term” which is the property of the thing being referred to, which may consist of one or more words. For example, the ISO 11179 name ‘***Acoustic Signal. Frequency. Measure***’ has the property term ‘***Frequency***’.
- A “Representation Term” which identifies allowable values for an element. This list is taken from an enumerated list of allowable representation types (see appendix A). For example, the ISO 11179 name ‘***Acoustic Signal. Frequency. Measure***’ has the “Representation Term” ‘***Measure***’.

The ebXML Technical Report, Naming Convention for Core Components ,provides 14 “rules” for constructing a proper data element names. Some considerations are:

- When the Representation Term and the Property Term are redundant, the property term is dropped, so ‘***Item. Identification. Identifier***’ becomes ‘***Item. Identifier***’.
- When an element describes an entire class of things (e.g., not a specific property of it), the Property Term may again be dropped, for instance ‘***Documentation. Identifier’***.
- An aggregate component shall have a Representation Term of ‘***details***’.

Note that ISO 11179 names **MAY** be made directly into XML component names:

- For XML Schema data types and XML attribute names.
- For XML element names when a business term cannot be found or agreed to.

The above discussion was taken from the initial set of specifications and technical reports produced by ebXML in May 2001. These initial documents formed a baseline form which OSIS and UN/CEFACT could jointly develop ebXML concepts. Appendix A provides more updated ISO 11179 and core component definition guidance that was taken from recent draft documents. This information **SHOULD** be used as guidance only, but may prove helpful.

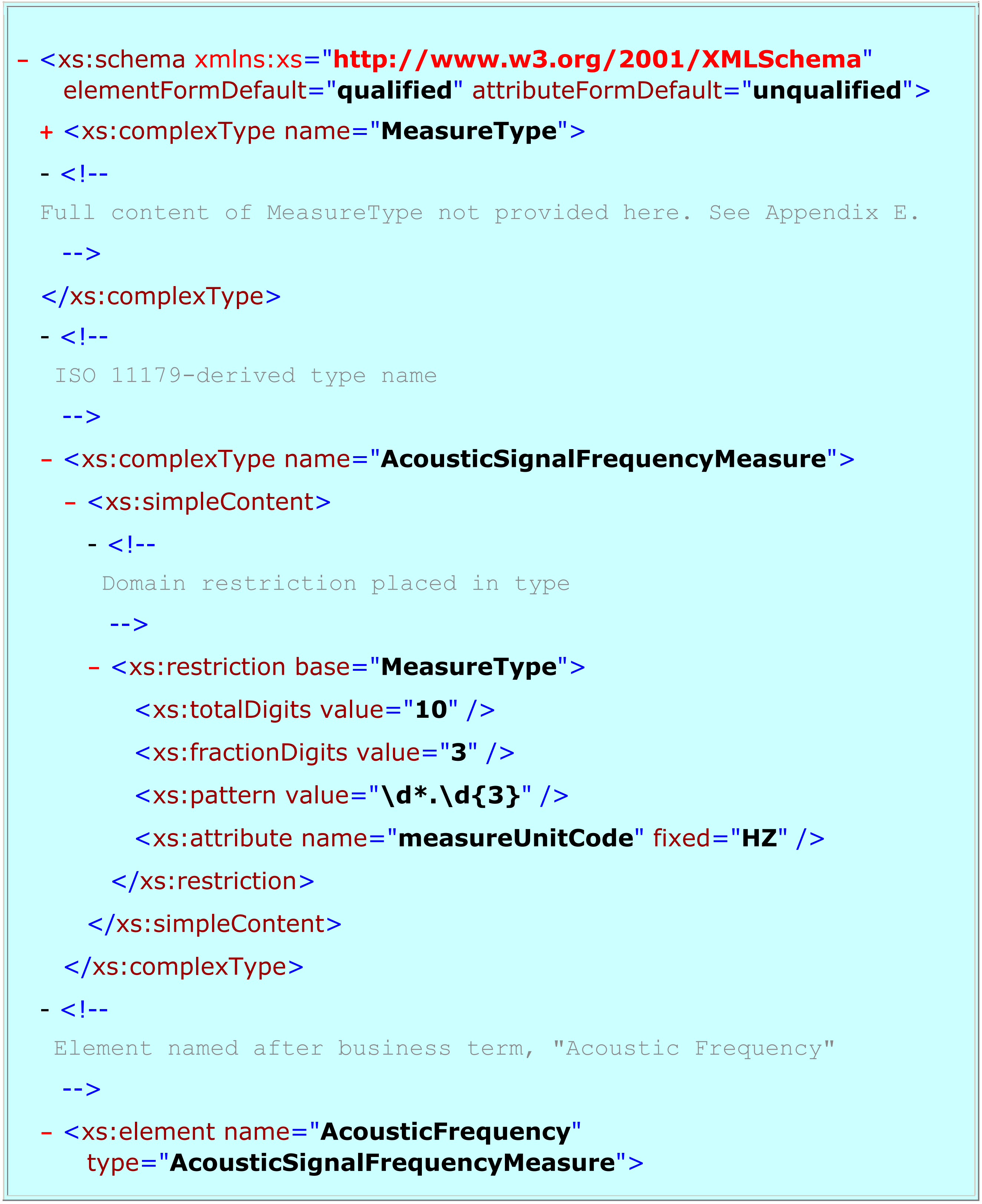
Example

A discovered component is considered not suitable if any of the above conditions are not met. Specifically, two examples of non-suitability may are:

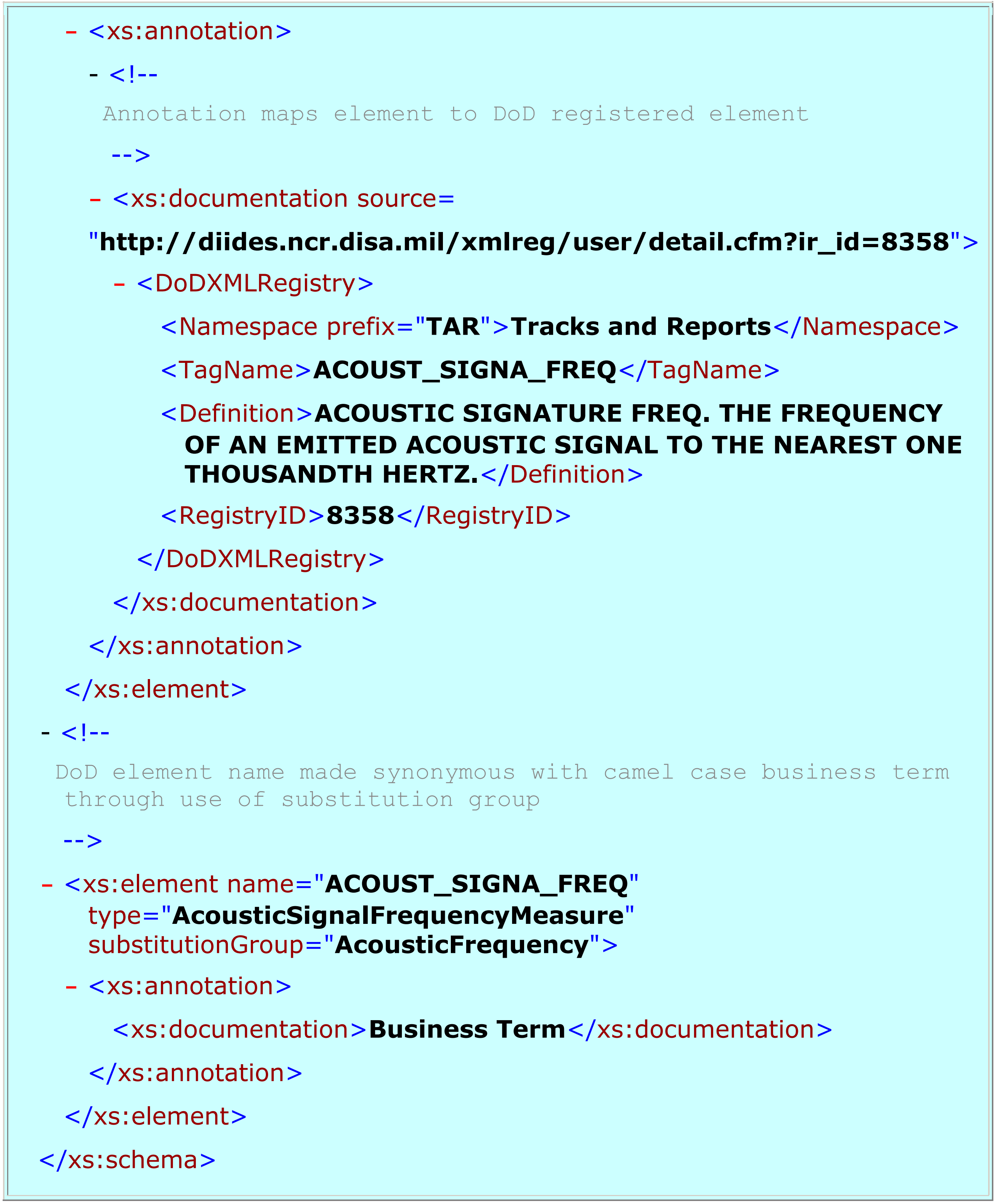
- The component is not suitable by virtue of naming convention differences. All other metadata (the definition, the domain range, etc. are acceptable).
- The component is not suitable because the required component is not an exact match to the component in the registry. For example, the required component’s domain range is outside the range of the registered component.

The following example is an excerpt from that provided in Appendix E.

```
<?xml version="1.0" encoding="UTF-8" ?>
```



```
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
  + <xs:complexType name="MeasureType">
  - <!--
  Full content of MeasureType not provided here. See Appendix E.
    -->
  </xs:complexType>
  - <!--
   ISO 11179-derived type name
    -->
  - <xs:complexType name="AcousticSignalFrequencyMeasure">
    - <xs:simpleContent>
      - <!--
       Domain restriction placed in type
        -->
      - <xs:restriction base="MeasureType">
          <xs:totalDigits value="10" />
          <xs:fractionDigits value="3" />
          <xs:pattern value="\d*.\d{3}" />
          <xs:attribute name="measureUnitCode" fixed="HZ" />
        </xs:restriction>
      </xs:simpleContent>
    </xs:complexType>
  - <!--
   Element named after business term, "Acoustic Frequency"
    -->
  - <xs:element name="AcousticFrequency"
      type="AcousticSignalFrequencyMeasure">
```



```
  - <xs:annotation>
    - <!--
     Annotation maps element to DoD registered element
      -->
    - <xs:documentation source=
    "http://diides.ncr.disa.mil/xmlreg/user/detail.cfm?ir_id=8358">
      - <DoDXMLRegistry>
          <Namespace prefix="TAR">Tracks and Reports</Namespace>
          <TagName>ACOUST_SIGNA_FREQ</TagName>
          <Definition>ACOUSTIC SIGNATURE FREQ. THE FREQUENCY
            OF AN EMITTED ACOUSTIC SIGNAL TO THE NEAREST ONE
            THOUSANDTH HERTZ.</Definition>
          <RegistryID>8358</RegistryID>
        </DoDXMLRegistry>
      </xs:documentation>
    </xs:annotation>
  </xs:element>
- <!--
 DoD element name made synonymous with camel case business term
  through use of substitution group
  -->
- <xs:element name="ACOUST_SIGNA_FREQ"
    type="AcousticSignalFrequencyMeasure"
    substitutionGroup="AcousticFrequency">
  - <xs:annotation>
      <xs:documentation>Business Term</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

7.1.3.1. Creating XML Element Names from Business Terms

Guidance

Developers **SHOULD** use business terms instead of ISO 11179 compliant names for element names when appropriate business terms exist; however, the underlying ISO 11179 name **SHOULD** be captured:

- If developing XML Schemas, a XML Schema data type **MAY** be created named after the ISO 11179 name converted to upper camel case (see section 7.1.3.2).
- If developing in DTDs, a fixed 'type' attribute **MAY** be created referencing the ISO 11179 name or an XML Comment **MAY** be used.

More than one business term may exist for a single element, such as when an acronym is commonly used instead of the full business name. If developing XML Schemas, extra synonymous business terms **MAY** be created and declared in the substitution group of the primary business term.

Acronyms and abbreviations MAY be part of a business term, but **MUST** conform to the guidance of Section 7.1.2.

Explanation

The ebXML deliverables define the concept of a Business Term. Business terms are commonly recognized words that are more appropriately used as XML element names, rather than the often-esoteric ISO 11179 conventions. Business terms improve the readability of schemas and instances, while the ISO 11179 names provide more precise and structured semantics. Both are desirable when business and technical personnel are working together to define XML grammars for the exchange of business information by IT systems.

This guidance may appear confusing because on one hand the creation of ISO 11179 names is recommended, but on the other, business terms are recommended for XML element names. The guidance is to define ISO 11179 standard names and capture those names through the use of the Schema "type" while retaining readability through using business terms as element names. Since the XML Schema is XML, those analysts interested in finding out, for instance, that "***National Stock Number***" is a business term for "***Federal Material Item. Identification. Details***" can look at the underlying type name of the <***NationalStockNumber***> tag.

Examples

See previous example and appendix E.

7.1.3.2. Creating XML Component Names from ISO 11179 Data Elements

Guidance

XML components **MAY** be named after ISO 11179 data element names:

- XML Elements **SHOULD** be named after ISO 11179 data element definitions when business terms do not exist.
- XML Attributes **SHOULD** be named after ISO 11179 data elements.
- XML Schema data types **MUST** be named after ISO 11179 data elements.

Explanation

ISO 11179 part 5 provides a standard for creating data elements. This standard employs a dot notation and white space to separate the various parts of the element and multiple words in a part respectively. In order to meet XML requirements for component naming, the ISO 11179 name must be converted to a Name Token.

The ISO 11179 part 5 standard provides a way to precisely create a data element definition and name. Using or referencing this name in a schema provides analysts with a better understanding of XML component semantics, while using business terms as element names improves readability.

Requiring types to conform to ISO 11179 conventions will facilitate automated analysis of schema components during any harmonization efforts.

The upper and lower camel case conventions are adopted from ebXML.

Example

In the example of Section 7.1.3, the type '***AcousticSignalFrequencyMeasure***' was created from the ISO 11179 standard data element '***Acoustic Signal. Frequency. Measure***'.

7.1.3.3. Choosing XML Component Names

Guidance

The selection of XML component names **MUST** be a thoughtful process involving business, functional, database, and system subject matter experts. In the schema design process, DON XML developers **MAY** use temporary or dummy XML component names while consensus is being reached on more carefully designed and defined names.

The creation and/or selection of XML component names and business terms:

- **MUST** involve domain subject matter experts (operational personnel, program managers, etc), functional data experts (database administrators, functional data manager, data modelers, etc…) and software developers. Application developers **MUST NOT** be left on their own to perform this function.

- **SHOULD** use definitions (from the DDDS, COE Data Emporium, MIL-STDs, or other credible standard data element definitions).
- **SHOULD NOT** create a Business Term just for the sake of having one; the existence and use of business terms **SHOULD** be determined by consensus of a community of users. When a business term is not apparent or does not exist, the ISO 11179 compliant name **MAY** be used as the XML component name instead.

Explanation

At a business level, the primary function of XML is to provide a meta-language for rigorously specifying the syntax of information exchange. Since information exchange involves multiple parties (at a minimum one sender and one receiver), XML specifies agreements between parties within a community of interest for a particular domain of information. XML itself does not require or provide a mechanism for defining semantics (precisely what is meant by a particular term); however, to achieve interoperability, both the syntax and semantics must be explicitly defined. The process of selecting proper component names and reaching agreements on the definitions is primarily a business function of XML and **MUST** involve all stakeholders. Frequently, application developers who are on the leading edge of technology will understand the benefits of XML and will implement it in IT systems before business personnel become involved. As a result, XML component names often are not useable by an entire community, seriously impeding widespread , understanding and therefore interoperability.

7.2. Schema Design

7.2.1. Schema Languages

Guidance

Only W3C-recommended languages **SHALL** be used within the DON for describing documents. Specifically, the DTD and the W3C recommended XML Schema language **SHALL** be used.

All activities developing data-oriented schemas in DTD syntax **SHOULD** plan on migrating to XML Schemas.

DON XML developers **MAY** elect to use DTDs for markup of data that is strictly document-oriented (paragraph, chapter, appendix...); however, the XML Schema language is preferred.

Explanation

Appendix H provides a business explanation for the adoption of XML Schemas over DTDs.

For activities that intend to migrate towards XML Schemas, an excellent free XML schema tutorial[vii] is available from www.xfront.com[viii]; it provides both detailed presentations and hands-on labs. Additionally, a series of XML Schema best practice papers[ix] is available. These papers provide more XML Schema development technical detail than is provided here.

Example

The DON guidance is to use XML Schema for creation of XML components; however the following are some example business case considerations for selecting DTD's over XML Schemas as the schema language them:

- An organization has an existing production XML implementation that meets all current and projected future requirements. It employs DTDs; and there is not sufficient funding in the budget to migrate to XML Schemas. In this case, there is no business case for investing in XML Schemas in the near future. Some points to note:
    - The application has achieved production status. It is not a pilot or demo.
    - There are no projected future requirements that would benefit from a Schema based approach. For data oriented applications, this situation is possible but unlikely.
- An organization's budget is so severely limited for migration to XML Schema such that investing in Schema development would impact the organizations ability to meet in-year operational requirements.
- An organization uses XML as a web-enabled version of SGML for markup of content that is primarily page-oriented (vice data oriented), and DTDs already exist for the page-oriented markup.

7.2.2. Recommended Schema Development Methodology

Guidance

DON XML developers **SHOULD** adopt the practice of developing schemas based on information exchange requirements identified via business process modeling. Information and process modeling and the XML schema creation process **SHOULD** be separate and distinct steps.

Schema development **SHOULD** take place as a team effort with functional data experts, business experts, program managers, and IT specialists all involved. The DON XML WG also strongly encourages collaboration among activities developing schemas within related information domains.

Conversely, schema development **SHOULD NOT** be solely the function of IT specialists. XML component names in general **SHOULD NOT** be taken directly from underlying relational database table and column names, unless the elements within that database have been named and created in accordance with a DON or DoD standard that represents concurrence by an entire Community of Interest (COI).

Explanation

The single most critical factor in creating logical, reusable schemas for information exchange in XML is the separation of the information modeling process from the schema creation process. Information should be modeled independently of creating a schema. This allows stakeholders to focus on creating logical, consistent representations of information, without getting distracted by the myriad of schema design options that have nothing to do with the information. Once an agreed to information model has been created, mapping rules from the model to a schema can be used or developed, which make schema creation straightforward. Just as this is the most important step, it is the most often neglected.

Typically, newly trained or inexperienced developers begin creating schemas on an ad hoc basis, without the involvement of business functional experts and without a carefully crafted information model that lends itself to expressing hierarchical, object-like relationships. Often, application developers working without management and functional involvement and without an appropriate model are tempted to create XML quickly and easily from relational database table and column names. XML components produced in this fashion have very terse, abbreviated and generally unreadable names, which are often not reusable by other systems or agreed to by the community of users.

The result of the actions in the above paragraph is inevitably a poorly-designed set of schemas with little reusability, extensibility, or readability; this translates into rework later at additional expense.

Because most uses of XML can be conceptualized as business processes in which communities of users share information, successful schema development should be based on analyzing, documenting, and reaching consensus on the *business processes,* the parcels of information (documents) exchanged in those processes, and the structure of a commonly-understood vocabulary / grammar for creating the documents.

The focus of XML schema and component development should be on creating XML languages that are understood by a community of stakeholders that engage in business processes together. In this context, the term *business process* is used in a larger scope than just business-to-business transactions (B2B) where products are bought and sold for money. Some examples:

- ♦ A supply activity wishes to make available, to its community, reference tables of code lists in an XML format. Here the process is consumer-to-application (C2A) / application-to-consumer (A2C) and application-to-application (A2A). A user (consumer) may request the table data via a web-browser (C2A); the activity receives the request and returns XML that is transformed to HTML (A2C). Also, an application may request and receive the same information in XML format via SOAP (A2A).

- A C4ISR application wishes to make air tasking order information, from messages, available on a publish-subscribe or broadcast basis to both operators and other C4ISR applications.
- A logistics activity wishes to store product data from an acquisition in a neutral format so that at some future point it can be parsed and read into any database for future processing by other activities needing it. In this case the process can be thought of as consumer-to-consumer (C2C), because the product data that is received by the acquiring consumer should be represented in an XML language that is understood by other consumers within the community.

Relational modeling languages, like IDEF1x, are appropriate for logical and physical enterprise data modeling of complex systems or data warehouses that will be implemented primarily by relational databases. However, modeling hierarchical, object-like relationships expressed by XML is more difficult in this language. Relational modeling focuses the efforts of the modeling exercise on the efficient representation of data as a set of normalized entities; this simplifies the process of creating relational databases but complicates the process of understanding the hierarchical nature of information, and it often hides or neglects critical object-like aspects of the domain.

XML is an information-sharing meta-language that is inherently hierarchical, lending itself to be better represented via graphical modeling languages that allow capture of object relationships vice key/key-reference relationships of normalized entities. The DON XML WG recommends that activities interested in capitalizing on XML as an information exchange medium take the time to learn UML. UML is rapidly becoming the de facto industry standard for system requirements analysis and business process and information modeling as well as software design. It provides a common language that business experts, managers and IT specialists can use throughout all phases of a system’s implementation (requirements discovery, analysis, business rules and workflow documentation, software design, and deployment).

Many data-modeling languages have an object orientation; however, products supporting the direct creation of XML DTDs and/or Schema from UML are becoming available, and the UN/CEFACT Electronic Business Transition Working Group[x] is standardizing a UML to XML[xi] mapping that will even further improve future tool support. By taking the time to create UML static structure models of information exchange requirements, schemas can be automatically generated and updated as standards and models evolve. This will ultimately drive down the cost of implementing XML based systems.

UML to XML tools are in their infancy. Due to lack of a standard, each tool does it differently at present. However, by taking the time to learn UML now, and beginning the process of creating information models in UML, DON activities will be well positioned to capitalize on future advancements.

Regardless of the modeling language chosen, it is useful to construct and use information and data models that are independent of XML-specific syntax. This will allow stakeholders involved in schema design to separate information-modeling decisions from XML design decisions.

The UN/CEFACT adopted Unified Modeling Methodology (UMM), based on UML, can be used for the process modeling; it will yield a business process model expressed in an XML syntax such that it can be universally understood and implemented. The DON XML WG expects to evaluate the UMM and other modeling methodologies for applicability to DON data domains for possible official adoption at a later date.

Examples

A proposed procedure for schema development is presented in Appendix E. It is non-normative, provided as an example only.

7.2.3. Capturing Metadata

Guidance

DON XML developers **SHOULD,** within reason**,** capture as much metadata as possible in a schema.

The schema language chosen (DTDs or XML Schema) will impact the amount of metadata that can be expressed and how well applications can access the metadata for processing.

- For DTDs, XML comments **MAY** be used to annotate the DTD with definitions and constraints, which the DTD syntax is unable to express.
- Alternatively, for DTDs, fixed attributes **MAY** be used to capture the metadata.
- For XML Schemas, metadata may be captured in a number of ways, as is discussed in the following sections. Guidance regarding the four primary ways of capturing metadata is as follows:
    - Domain value restrictions **SHOULD** be captured by the use of built-in Schema data types, the construction of custom data types, the assignment of enumerations to XML component values, the use of regular expressions, and minimum / maximum value constraints.
    - Metadata regarding the structure and cardinality of components **SHOULD** be captured by expressing element order as either a (set of) choice(s), an ordered sequence, or unordered. Additionally, the exact number of times an element can, or must, be repeated **MAY** be specified.
    - Logical relationships or relationships to existing data dictionaries and models (such as the DDDS, ebXML core components, or COE Reference Data Sets) may be expressed by the use of types or Schema annotations.

- ➢ An element's definition, sources of definitions or code lists, version information, and other metadata **MAY** be captured by the use of Schema annotations.

- ♦ Developers **MAY** consider the creation of a verbose semantic schema and a compact schema strictly for document validation purpose.
- ♦ Alternatively, schema documentation and annotations **MAY** be provided by creating a schema guide that is URL-accessible and referenced in the header of the schema. Tools such as XML Spy 4.x provide excellent documentation generation capabilities that can partially automate this process.

Explanation

The schema is more than just a document structure validation tool. The XML Schema language, in particular, has a rich feature set for capturing extra metadata that can provide:

- ♦ Data element definitions through the use of annotations
- ♦ Detailed domain value constraints
- ♦ Logical data element pedigree through the use of annotations and types.

By capturing this metadata, the schema becomes an interoperability tool, because analysts can read it and understand what the various XML components mean and where they are derived from. Several sources of metadata exist that can be used to derive XML components; these include:

- ♦ The DoD XML Registry[xii]
- ♦ The initial set of ebXML core components (see the ebXML Technical Reports[xiii] on Core Components)
- ♦ The DDDS
- ♦ The COE Data Emporium Reference Data Sets[xiv].
- ♦ Various Military Standards (MIL-STD-6040[xv], 6011, 6016, etc.)
- ♦ Various commercial standards (ISO, ANSI, IEEE etc.)

With the exception of the DoD XML Registry, the sources named do not provide readily reusable XML component names; however, they do provide agreed to, reusable data element definitions.

A fully documented XML Schema may be quite verbose. Such “semantic” Schemas can provide critical insight to analysts and improve interoperability by making use of the information in the Schema. However, they contain much more information than is really necessary for document structure validation. A “compact” Schema that is equivalent to the “semantic” Schema may be quickly built for validation purposes. Having both a full “semantic” Schema and a “compact” schema may be appropriate for activities wishing to provide extensive Schema annotations, or underlying type relationships while having a smaller schema used strictly for validation.

A *schema guide* document that fully defines and explains each component in schema and the schema's logical structure is an alternative to creating a fully documented semantic schema.

Example

Appendix E provides an example that combines several of the concepts discussed so far, including capturing definitions and relationships.

7.2.3.1. Application Specific Metadata

Guidance

Application-specific metadata (such as SQL statements or API calls) **MUST NOT** be included in instances or schemas that describe payloads of information to be exchanged between applications.

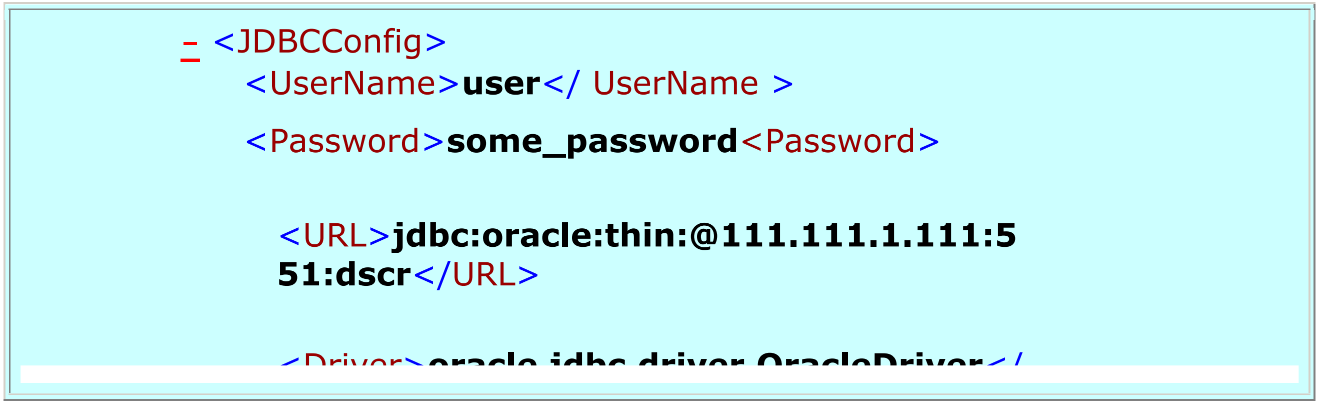
Conversely, XML **MAY** be used to capture application specific metadata and initialization parameters so long as the XML instance is separate from information payload XML.

Explanation

Including application-specific metadata in an instance unnecessarily clutters the document, increases bandwidth requirements, and is only useful to one application. However, an emerging use of XML to capture application specific initialization parameters (in place of the traditional "ini" files) is very useful. The only prohibition is that application initialization XML and XML used to expose or exchange business information must be physically separate documents.

Example

Example of an XML document that provide JDBC initialization parameters to an application



```
- <JDBCConfig>
    <UserName>user</ UserName >
    <Password>some_password<Password>
      <URL>jdbc:oracle:thin:@111.111.1.111:5
      51:dscr</URL>
      <Driver>oracle.jdbc.driver.OracleDriver</
```

```
          Driver>
      </JDBCConfig>
```

Example of an XML document carrying a “payload” of business information:

```
      - <UnitLatitude MeasureUnitCode=”DEG”>30.500
        </ UnitLatitude >
```

1

7.2.3.2. Capturing XML Component Definitions

Guidance

DON XML developers **MUST** document XML element and XML Schema type definitions through XML comments, XML Schema annotations, a schema guide, or a data dictionary. These definitions **SHOULD** be related to underlying ISO 11179 data element definitions.

Definitions **SHOULD** be brief and when possible **SHOULD** be taken from existing standard data element definitions, such as those provided by the DDDS, ebXML Core Components, COE Reference Data Sets, or other Military Standards (MIL-STD-6040, 6011, 6016, etc.)

Definitions **SHOULD** contain URLs or other pointers to the definition’s source, so that analysts can look up additional information.

Developers **MAY** extend the XML Schema annotation <***xsd:documentation***> tag by further marking up information provided with custom tags. No standards for this yet exist; however, the general guidelines of this document should be followed, and custom metadata tag names should follow the naming convention of the source data dictionary.

Developers **MAY** elect to publish schema documentation in a separate *schema guide*; however, if this option is chosen, the schema must be URL-accessible and referenced in the schema header.

Explanation

Many activities in the DON are rapidly developing schemas as part of initiatives such as TFWeb. Mandating that schema developers take the time to provide element and Schema type definitions will facilitate identifying commonalities and reusable components. Furthermore, it will start to enforce some rigor and thought in the creation of XML components, as business and technical experts come together to create definitions for components and map their context specific elements back to applicable DON and DoD enterprise data standards.

Section 7.4 provides guidance on use of XML elements vice attributes. It is the DON XML WG's recommendation that attributes be minimized, and only used to provide supplementary metadata necessary to understand the business value of an XML element. By adopting this convention, and that of naming attributes in camel case according to ISO 11179 conventions, attributes will be reasonably self-explanatory and therefore not require a definition in most cases.

Example

Appendix E provides a consolidated example of capturing definitions in XML Schema.

Examples Section 6.1.2 also illustrates these concepts.

7.2.3.3. Enumerations and Capturing Code Lists

Guidance

DON XML schema developers **SHOULD** use XML Schemas to express enumeration constraints on XML element and attribute values, when such enumerated lists are of reasonable length and when code lists are considered stable (not likely to change frequently).

The decision to explicitly enumerate in a schema **SHOULD** be made by program managers based on the resulting size of the schema, bandwidth availability, and validation requirements.

Code lists, from which enumerations are taken, **SHOULD** be referenced by URI or other pointers so that analysts can look up code values.

Explanation

The DoD frequently represents data element values as codes rather than as free text. Codes are much easier for an application to understand and process because they are taken from a finite list of possible values, each with agreed-upon semantics. Application developers create software to execute actions based on those code definitions and a specified set of business rules. XML can be used to exchange data that uses codes to abbreviate information, and the schema can be used to provide metadata about codes and their associated definitions (reference tables). Again, the way this is accomplished depends on the schema language chosen, with XML Schemas offering the most functionality. Capturing a reference to a list of valid codes and code values will greatly enhance implementations and allow future analysis to identify standard code reference tables. However, for code lists that historically change frequently, a URI pointer to the authoritative code list source is preferable.

Example

A DTD example of an element taken from the MIL-STD-6040 (USMTF) with an enumerated set of possible values and an XML comment referencing the source of the code definitions.

```
<!ELEMENT Casualty EMPTY>
<!ATTLIST  Casualty casualtyCategoryCode (1 | 2 | 3 | 4 )
#REQUIRED>
<!-- casualtyCategoryCode
Definition: A CATEGORY DENOTING THE EFFECT OF A CASUALTY ON A
UNIT'S PRIMARY AND/OR SECONDARY MISSION AREAS.
Source: MIL-STD-6040 Baseline 2001 FFIRN 1207 FUDN 0001 -->
```

7.3. Document Annotations

Guidance

DON XML schema developers **MUST** provide carefully thought out comments within schemas and stylesheets, which provide basic information necessary to use and understand the document.

In general, Instances **SHOULD NOT** be documented; however, there may be situations where it is appropriate.

Explanation

Just as it is good programming practice to document application code using a coding standard, it is important that XML schemas and stylesheets be well documented in a standard fashion. The following paragraphs provide some recommended guidance.

The simplest way to express annotations is through the use of XML Comments. Comments can be inserted anywhere in an XML document after the XML Declaration.

XML Schema annotations provide a more flexible, extensible way to document Schemas as illustrated by many examples in this document.

7.3.1. Document Versioning

Guidance

Version information for instances, schemas, and stylesheets **MUST** be available via document annotations (XML comments or Schema annotations) or through built in attributes where the W3C syntax allows.

Explanation

Having a schema's version number available to developers will assist in creating implementation that will maintain backward compatibility. Version information is also necessary for stylesheets in order to determine which version of a stylesheet correctly transforms an instance that conforms to a version of a schema.

7.3.1.1. Versioning DTDs

Guidance

DTD version information **SHOULD** be captured as an XML comment in the header of the DTD, and **MAY** be captured as a fixed attribute of the root element or **MAY** be appended to the DTD file name to uniquely identify it.

Another option is to append a version number to the DTD name, thus uniquely identifying it from previous versions.

Explanation

DTDs offer two means of documenting version number. The most straightforward is to put the DTD version number in the header XML comment. A second method is to declare a fixed schema version attribute to the XML Root Element. This will make the version generally available to applications via an API call.

Uniquely identifying a DTD name by appending a version will prevent applications that process a different version of the same schema from validating the instance. This may or may not be desirable. However, since DTD do not have a built in version attribute like XML Schema, this is one strategy that will allow an application to catch version mismatch.

A best practice for DTD versioning has not been identified; therefore developer feedback is encouraged.

Example

```
<?xml version='1.0' encoding='UTF-8' ?>
<!ELEMENT root EMPTY>
<!ATTLIST root  schemaVersion CDATA  #FIXED '1.0' >
```

Example of a versioned DTD name: “***rootV1.1.dtd***”

Providing version information in an XML comment in the header of a schema is discussed in Section 7.3.2.

7.3.1.2. Versioning XML Schemas

Guidance

XML Schemas **MUST** include a version using the 'version' attribute of the XML Schema specification.

Explanation

The schema header as discussed in Section 7.3.2 provides a uniform method to capture a consistent body of information required for a schema. However, developers can make version information more easily available to applications through the use of the version attribute as shown in the example.

Example

Example of using the version attribute of and XML Schema to capture schema version information:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified" version="1.0" >

  ...

</xsd:schema>
```

7.3.1.3. Versioning Stylesheets

Guidance

A stylesheet **MUST** contain both its own version number (by using the built-in version attribute of the XSLT language) and references to the name and versions of the schema that describe instances upon which the stylesheet performs correctly.

Explanation

Tracking versions of stylesheets is very important because a new version of a stylesheet may or may not correctly transform an instance conforming to an old version of a schema. Explicitly asserting in a stylesheet which versions of a schema are supported will alleviate potential interoperability issues as implementations evolve.

Example

See example provided in Appendix F.

7.3.2. Headers

Guidance

To promote interoperability, every schema, stylesheet, or instance **MUST** contain some basic metadata.

The following metadata **SHOULD** be provided:

7.3.2.1. Schema :

- Schema Name
- DoD Namespace(s)
- Navy Functional Data Area *[Ed Note: insert URL to DMI document that defines]*
- URL to most current version
- For XML Schema, other Schemas imported or included to include DoD Namespace and version Schema file name, and URL.
- For DTD, external entities referenced to include DoD Namespace and version (in the case of parameter entities that are modular DTDs)
- A description of the purpose of the schema
- The name of the application or program of record that created and and/or manages the schema
- The version of the application or program of record
- A short description of the application interface that uses the description. A URL reference to a more detailed interface description may be provided
- Developer point of contact information to include activity, name and email
- A change history log that includes change number, version, date and change description

7.3.2.2. Stylesheets:

- Stylesheet Name
- A list of schemas and XSL processors that the stylesheet have been tested against
- The DoD Namespace where the stylesheet is registered
- Navy Functional Data Area of the application that makes use of the stylesheet
- URL to most current version
- Other stylesheets imported to include name and URL
- A description of the purpose and function of the stylesheet

- Application or program of record (with version) responsible for developing and maintaining the stylesheet
- Developer point of contact information to include activity, name and email
- A change history log that includes change number, version, date and change description

7.3.2.3. Instances

- The name and URL of the schema that validates, and the stylesheet (if any) that correctly transforms it, if these are not specified already as part of the instance.

Explanation

Other interested parties must be able to read a document and understand how to implement it or use information from it. Much of the information captured in a header XML comment can be better made available to applications through the use of fixed attributes or XML Schema annotations. However, having a consistent set of header information in a consistent location in an XML document will promote better configuration management and interoperability as methods for making this information available to applications are standardized. While examples are provided that show the above information captured in a single comment after the XML declaration, this should not discourage innovative developers from providing the same information as Schema annotations (possible with custom markup inside a <***xsd:documentation***> tag.) Some information may also be captured as fixed attributes if developing in DTDs, as illustrated by previous examples.

Example

Appendix F provides non-normative examples of document headers.

7.4. Attributes vs. Elements

Guidance

The use of attributes **SHOULD** be carefully considered . Attributes, if used, **SHOULD** provide extra metadata required to better understand the business value of an element.

Some additional guidelines are:

- Attribute values **SHOULD** be short, preferably numbers or conforming to the XML Name Token convention. Attributes with long string values **SHOULD NOT** be created.
- Attributes **SHOULD** only be used to describe information units that cannot or will not be further extended, or subdivided.

- Information specific to a single application or database **MUST NOT** be expressed as values of attributes (see Section 7.2.3.1)
- Attributes **SHOULD** be used to provide metadata that describes the entire contents of an element. If the element has children, any attributes **SHOULD** be generally applicable to all the children.

Explanation

One of the key schema design decisions is whether to represent an information element as an XML element or attribute. Once an information element has been declared an attribute, it cannot be extended further; for this reason and to promote better uniformity within the DON, the use of attributes is not encouraged.

Example

In Example 1, the code KTS (for knots) provides extra metadata required to understand the 'business value' of the element – 600. It answers the question, "600 what?"

In the other examples, several appropriate ways of expressing coded values are illustrated.

Example 1:

```
<TargetVelocityMeasure measureUnitCode=”KTS”>600</
    TargetVelocityMeasure>
```

Examples of inappropriate attribute usage

Example 2:

```
<TargetVelocity measure=”600” measureUnitCode=”KTS”/>
```

Example 3:

```
<CasualtyCategoryCode definition=”[TRAINING ACTIVITY ONLY]
    EQUIPMENT CASUALTY EXISTS BUT WILL NOT IMPACT
    TRAINING WITHIN 30 DAYS.”> 1</CasualtyCategoryCode>
```

In example 2, both the business value and descriptive metadata are attribute values. This provides no mechanism for applications to determine which piece of information describes the other. In example 3, the attribute is used to provide a verbose definition while the code value is the element contents; because XML parsers normalize white space in attribute values, attributes are inappropriate for use in this manner.

8. Points of Contact

DON XML WG Government Lead:

Michael Jacobs, Jacobs.Michael@hq.navy.mil , (703) 601-3594

DON XML Technical Lead and Editor:

Brian Hopkins, xosys@sbcglobal.com, (858) 793-7369

9. Document History

Initial DON XML Developer's Guide 29 October

This document is the initial XML Development guidance promulgated by the DON XML WG; it represents an abbreviated version of the full 9 October Consensus Draft titled "XML Developers Guide – 9 October". It did not go through the full consensus process as described by the DON XML WG Operating Guidelines and therefore does not represent a consensus of the entire team. This document was produced by key individuals of the DON XML Technical Team and Steering Group in order to support the Task Force Web (TFWeb) pilot project milestones.

Initial DON XML Developer's Guide V1.1

Still titled "Initial," this document represents the first minor revision to the 29 October Developer's guide. While it is only a "minor" revision, the changes are significant. The document should be review thoroughly.

Summary of structural and global changes:

- Section 3 and 4 reorganized and reworded. Second paragraph of Section 3 removed as was redundant.
- Section 7 (DoD XML Registry) moved to Section 5, renumbered all other sections.
- Added line numbers.
- Added Appendix H to provide a business explanation of the advantages of XML Schemas over DTDs. Removed explanation from Section 7.
- Changed *COE* to *DoD* in all references to Registry and Namespaces.

- Introduced new term, Voluntary Consensus Standard. See Appendix G. This term is used extensively through document to replace references to OASIS, BizTalk, RossettaNet, etc…
- Removed the Word “Initial” from the title.

Summary of Significant Guidance Changes

- Section 3 – Terminology and Conventions (V 1.0 section 4)
    - Moved RFC 2219 reference here.
- Section 4 – Implementation Requirements (V1.0 section 3)
    - Reorganized guidance into 4 subsections, 2 of which are new. Section 4.1 specifically establishes the requirements level of the document as guidance, 4.2 specifically names the program manager as the final conformance authority, and 4.4 provide additional clarification as to the guidance applicability.
    - Specifically gives this document precedence over other Navy guidance for matters pertaining to XML.
- Section 5 – DoD XML Registry
    - Reuse of Voluntary Consensus Standards XML components is mentioned first.
    - Additionally, emerging DoD XML policy is referenced that will require registration of VCS tags used.
- Section 6 – Recommended XML Specifications
    - Guidance changed to clarify the precedence of accredited standards bodies (like IEEE, UN/ECE, ISO, and ANSI), the W3C, and Voluntary Consensus Standards bodies like OASIS, RossettaNet and others.
    - OASIS is given precedence equivalent to accredited standards bodies.
    - Precedence is given to W3C final recommended technical reports relating to XML.
    - So that W3C work does not gain “instant credibility”, W3C working drafts must be at the second stage before being considered over other competing standards.
    - Structure of guidance reoriented to be centered on kind of application (production, pilot, demo) vice W3C status.
    - Added guidance on SOAP and SAX.
    - Provide clarification in explanation section of relationship of ebXML to other organizations.

  - Non-W3C draft specification given same status as W3C Working Draft level products.
- Section 7 – XML Conventions
  - Section 7.1 – XML Components
    - 7.1.2 Usage of Acronyms and Abbreviations: Changed guidance on acronyms and abbreviations to remove the prohibition on use of abbreviations. Added a program manager’s discretion clause and extra explanation. Basis for usage of abbreviations should be on belief that it will add to understanding.
    - 7.1.3. XML Component Selection and Creation: New section added to replace V1.0 section 6.1.3. Developed more detailed guidance on reuse of XML component from DoD XML registry including criteria for suitability for reuse. New sections with clarified old guidance, and additional new guidance. Added several paragraphs to the beginning of this section discussing priority of commercial, DoD and DON XML component reuse and creation. Order of precedence is commercial, DoD, then DON. Among commercial, precedence is given to W3C, the accredited standards bodies (including OASIS), then other Voluntary Consensus Standards.
      - 7.1.3.1 Creating XML Component Names from Business Terms.
      - 7.1.3.2. Creating XML Component Names from ISO 11179 Data Elements: Separated section on creating XML component names from ISO 11179 data elements and added more detail. Removed prohibition on using “Details” in element or type names.
      - 7.1.3 Choosing XML Component Names – Bulk of V1.0 Section 6.1.3 is here.
  - 7.2 Schema Design
    - 7.2.1 Schema Languages – Added in guidance and examples of when a DTD may be the appropriate schema language. Removed lengthy explanation, moved to Appendix H and replace wording with business explanation taken from draft Universal Business Language (UBL) documents.
    - 7.2.3 Capturing Metadata
      - 7.2.3.1 Application Specific Metadata – Banned all application specific metadata from payload instance of XML, but recommended use of XML as a format for storing application initialization parameters, as long as this was done separately from payload XML.

- 7.3 Document Annotations
    - 7.3.1 Document Versioning
        - 7.3.1.1 Versioning DTDs – Introduces option to append version information to the end of a DTD name.
        - 7.3.1.2. Versioning XML Schemas: Corrected example to illustrate the use of the built in ‘version’ attribute of the XML Schema root element.
        - 7.3.1.3. Versioning Stylesheets: Removed example.
- 7.4. Attributes vs. Elements: Removed references to and examples relating to using attributes to capture code definitions. Changed “Examples” to include one “good” and two “poor” uses of attributes.

10. Appendices

The following appendices are presented in draft form. They represent the understanding and opinion of the editor and are not the consensus of the DON XML WG. They are provided, as is, and are non-normative.

Appendix A – ebXML and the eBTWG

Description

ebXML was a 18-month international project sponsored jointly by OASIS[xvi] and UN/CEFACT[xvii] that ended in May, 2001 with the delivery of several specifications, technical reports and white papers available at www.ebxml.org/specs . The ebXML deliverables defines an architecture with two distinct views. The Functional Service View (FSV) defines:

- ♦ Functional capabilities
- ♦ *Business Service Interfaces*
- Protocols and *Messaging Services.*

In other words, the FSV consists of specifications and standards that describe how an ebXML compliant system will physically operate to include interfaces, protocols, and registry/repository operations.

The Business Operational View (*BOV)* addresses:

a) The semantics of business data in transactions and associated data interchanges

b) The architecture for business transactions, including:

- ♦ Operational conventions
- ♦ Agreements and arrangements
- ♦ Mutual obligations and requirements

The BOV work focused on two areas. The first focus was on creating a methodology by which business processes can be modeled as orchestrated collaborations between business partners who exchange payloads of information (which may be XML documents). The UMM was chosen as the modeling methodology and a BPSS was created. Second, the BOV work focused on creating a methodology for creating reusable components – process components which can be used to build complex business process models, and information components which can be used to construct business documents as payloads of ebXML messages. Some of the ebXML technical reports discuss the concept of core components as universal, domain independent information entities defined in an XML-neutral syntax. This is significant because the ebXML authors intentionally did not address how components (core and domain specific) should be used to produce business documents (in XML). According to the ebXML architecture, ebXML components exist as registered objects within an ebXML registry/repository system; the work of defining production rules for creating XML payloads from registry entries was deferred. This decision has drawn sharp criticism from some; however, it makes sense. The ebXML strategy was to address how to represent information (semantics and context) independently of how it is syntactically expressed as an XML document; consequently the ebXML technical reports on core components adopt the

ISO 11179 naming convention for creation of ***dictionary entries*** for ***information entities***. They do not specify how to create XML component names for schemas describing business documents containing payloads of information.

The ebXML deliverables provide a basis for future work required to make the vision of global interoperability a reality. OASIS and UN/CEFACT agreed to divide that work between them with OASIS assuming responsibility for the FSV aspects while UN/CEFACT took on the BOV portion. Since that time, UN/CEFACT has established the Electronic Business Transition Working Group (eBTWG[xviii]),

> *...for the purpose of continuing the UN/CEFACT's role in pioneering the development of XML standards for electronic business. The group was formed to build on the success of the earlier ebXML Joint Initiative between UN/CEFACT and OASIS, which delivered its first set of specifications in May 2001.*

One of the key deliverables of this group will be a final Core Component Specification that will combine and further refine the ebXML Core Component Technical Reports[xix].

The rest of the information presented in this appendix is taken from the deliverables of the ebXML project. These documents are works in progress. They may be useful in selecting data element and XML component names; however, developers must and should expect the rules and specifications presented here to evolve rapidly.

ebXML Naming Rules

Quoted[4] from the ebXML Technical Architecture[xx], Section *4.3 Design Conventions for ebXML Specifications:*

> “In order to enforce a consistent capitalization and naming convention across all ebXML specifications "Upper Camel Case" (*UCC*) and "Lower Camel Case" (*LCC*) Capitalization styles **SHALL** be used. *UCC* style capitalizes the first character of each word and compounds the name. *LCC* style capitalizes the first character of each word except the first word.

[4] Copyright © ebXML 2001. All Rights Reserved.

“This document and translations of it MAY be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation MAY be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself MAY not be modified in any way, such as by removing the copyright notice or references to ebXML, UN/CEFACT, or OASIS, except as required to translate it into languages other than English.”

Appendix A XML Developer's Guide V1.1 – 1 May 2002

1) ebXML DTD, XML Schema and *XML* instance documents **SHALL** have the effect of producing ebXML *XML* instance documents such that:

- Element names **SHALL** be in *UCC* convention (example:

  <UpperCamelCaseElement/>).

- Attribute names **SHALL** be in *LCC* convention (example: <UpperCamelCaseElement lowerCamelCaseAttribute="Whatever"/>)...

3) General rules for all names are:

- Acronyms **SHOULD** be avoided, but in cases where they are used, the capitalization **SHALL** remain (example: XMLSignature).
- Underscore (_), periods (.) and dashes (-) **MUST NOT** be used (don't use: header.manifest, stock_quote_5, commercial-transaction, use HeaderManifest, stockQuote5, CommercialTransaction instead)."

The following are component-naming rules as quoted from the technical report, Naming Convention for Core Components[xxi] Section 5.2. They are based on the ISO 11179 Part 5 draft specification. In reading these understand that:

- Since the publication of this report, the eBTWG has changed "representation type" to "representation term":
- These rules apply to creation of ebXML "core components" but may be used in the creation of DON specific elements as well.
- These initial rules are in being incorporated into the eBTWG's Core Components Specification, which is being developed by the Core Component project team. Developers may choose to use the rules specified in the draft Core Components Specification rather than these. When that document reaches final status, this appendix will be updated accordingly. For now the May, 2001 Core Component Naming Convention rules as specified by the initial ebXML project are provided for reference.

Rule 1: The Dictionary Entry Name shall be unique and shall consist of Object Class, a Property Term, and Representation Type.

Rule 2: The Object Class represents the logical data grouping (in a logical data model) to which a data element belongs" (ISO 11179). The Object Class is the part of a core component's Dictionary Entry Name that represents an activity or object in a context.

An Object Class may be individual or aggregated from core components. It may be named by using more than one word.

Rule 3: The Property Term shall represent the distinguishing characteristic of the business entity. The Property Term shall occur naturally in the definition.

Rule 4: The Representation Type shall describe the form of the set of valid values for an information element[5]. It shall be one of the terms specified in the "list of Representation Types" as included in this document.

Note: If the Representation Type of an entry is "code" there is often a need for an additional entry for its textual representation. The Object Class and Property Term of such entries shall be the same.

(Example : "***Car. Colour. Code***" and "***Car. Colour. Text***").

Rule 5: A Dictionary Entry Name shall not contain consecutive redundant words. If the Property Term uses the same word as the Representation Type, this word shall be removed from the Property Term part of the Dictionary Entry Name.

For example: If the Object Class is "goods", the Property Term is "delivery date", and Representation Type is "date", the Dictionary Entry Name is 'Goods. Delivery. Date'.

In adoption of this rule the Property Term "Identification" could be omitted if the Representation Type is "Identifier".

For example: The identifier of a party ("Party. Identification. Identifier") will be truncated to "Party. Identifier".

Rule 6: One and only one Property Term is normally present in a Dictionary Entry Name although there may be circumstances where no property term is included; e.g. Currency. Code.

Rule 7: The Representation Type shall be present in a Dictionary Entry Name. It must not be truncated.

Rule 8: To identify an object or a person by its name the Representation Type "name" shall be used.

Rule 9: A Dictionary Entry Name and all its components shall be in singular form unless the concept itself is plural; e.g. goods.

Rule 10: An Object Class as well as a Property Term may be composed of one or more words.

Rule 11: The components of a Dictionary Entry Name shall be separated by dots followed by a space character. The words in multi-word Object Classes and multi-word Property Terms shall be separated by the space character. Every word shall start with a capital letter

Rule 12: Non-letter characters may only be used if required by language rules.

[5] The term 'information element' is used generically in the same context as the term data element, and should not be confused with XML Elements. An information element (or entity as ebXML refers to them) can be expressed as any of several XML components (XML Elements, attributes, or XML Schema data types).

Rule 13: Abbreviations, acronyms and initials shall not be used as part of a Dictionary Entry Name, except where they are used within business terms like real words; e.g. EAN.UCC global location number, DUNS number [see section 5.1.2 Usage of Acronyms and Abbreviations]

Rule 14: All accepted acronyms and abbreviations shall be included in an ebXML glossary [read, "...included in the element definition in the schema annotation, see section 5.1.2]."

Representation Terms

The following extract is provided from a 12 October 2001 draft of the eBTWG core component specification. It is provided for information only: Here *Representation Term* is used vice the earlier *Representation Type* initially used in the ebXML technical reports.

Table 6-3 Representation Terms

Representation Term	Definition	Links to Core Component Type
Amount	A number of monetary units specified in a currency where the unit of currency is explicit or implied.	Amount. Type
Code	A character string (letters, figures or symbols) that for brevity and / or language independence may be used to represent or replace a definitive value or text of an attribute. Codes usually are maintained in code lists per attribute type (e.g. colour).	Code. Type
Date	A day within a particular calendar year (ISO 8601).	Date Time. Type
Date Time	A particular point in the progression of time (ISO 8601).	Date Time. Type
Graphic	A diagram, graph, mathematical curves, or similar representation	Graphic. Type

Representation Term	**Definition**	**Links to Core Component Type**
Identifier	A character string used to identify and distinguish uniquely, one instance of an object within an identification scheme from all other objects within the same scheme. [Note: Type shall not be used when a person or an object is identified by its name. In this case the Representation Term "Name" shall be used.]	Identifier. Type
Indicator	A list of two, and only two, values that indicate a condition such as on/off; true/false etc. (synonym: "Boolean").	Indicator. Type
Measure	A numeric value determined by measuring an object. Measures are specified with a unit of measure. The applicable unit of measure is taken from UN/ECE Rec. 20.	Measure. Type
Name	A word or phrase that constitutes the distinctive designation of a person, place, thing or concept.	Text. Type
Percent	A rate expressed in hundredths between two values that have the same unit of measure.	Numeric. Type
Picture	A visual representation of a person, object, or scene	Picture. Type
Quantity	A number of non-monetary units. It is associated with the indication of objects. Quantities need to be specified with a unit of quantity.	Quantity. Type
Rate	A quantity or amount measured with respect to another measured quantity or amount, or a fixed or appropriate charge, cost or value e.g. US Dollars per hour, US Dollars per EURO, kilometre per litre, etc.	Numeric. Type
Text	A character string generally in the form of words of a language.	Text. Type
Time	The time within a (not specified) day (ISO 8601).	Date Time. Type

Representation Term	Definition	Links to Core Component Type
Value	Numeric information that is assigned or is determined by calculation, counting or sequencing. It does not require a unit of quantity or a unit of measure	Numeric. Type

The following representation terms apply to aggregate Core Components or Core Component types.

Table 6-4 Other Representation Terms

Representation Term	Definition	Links to Core Component Type
Details	The expression of the aggregation of Core Components to indicate higher levelled information entities	Not Applicable
Type	The expression of the aggregation of Core Components to indicate the aggregation of lower levelled information entities to become Core Component Types. All Core Component Types shall use this Representation Term	Not Applicable
Content	The actual content of an information entity. Content is the first information entity in a Core Component Type	Used with the content components of Core Component Types

The ebXML core components technical reports require that name of "aggregate information entities" use the special representation type, 'details'. DON XML developers may omit the term 'details' from the end of tag names when XML element names are generated from the ISO 11179 name. For example, the ISO 11179 data element name '***Address. Details'*** would be represented in the XML instance as <***Address***>; in the XML Schema that describes the instance, the element ***Address*** would be created from the ISO 1179 derived Schema type ***AddressDetails***.

The Representation Terms provided by ISO 11179 may not be adequate for a number of engineering, scientific and operational concepts. In these cases, use of other term names temporarily, such as until the list of types is expanded, **MAY** be considered; however, do this with caution.

Appendix B – Schema Development

Possible Schema Development Procedure Summary

The following is presented as a possible procedure for developing schema. It does not represent the consensus of the DON XML WG; rather, it is presented for your consideration and feedback. It is purely developmental; all or none of it may be found useful.

STEPS

In creating XML components according to these conventions, try the following :

Step 1. Analyze the business processes in which your application will exchange, use or store information. Understand who the consumers (both human and machine) of the information your application provides are. The DON XML WG recommends the use of the UMM and UML for this process; however, any model that provides a basic understanding of how information will be exchanged across system boundaries (application to application, application to human, or human to application) can provide a basis for development as more rigorous modeling techniques, such as the UMM, are learned. The business process modeling should identify and name actors (persons, organizations, or systems) that participate in the process. The roles that each actor plays should also be identified and named. It is important to separate the name of the actor from the name of the role because often the same actor will participate in multiple roles within a process.

Step 2. Based on the information exchange requirements identified in step 1, spend the time to model the data in each document that will be exchanged within the processes defined in step 1. DON XML strongly recommends using the Unified Modeling Language (UML) to conduct the modeling. Several efforts are underway to create production rules by which UML models can be used directly to generate XML documents. An excellent online resource is xmlmodeling.com.

Step 3. Look for previously developed XML components that can be reused, either in the DoD XML Registry or schema developed by commercial consortia (Appendix D provides references).

Step 4. Create the ebXML/ISO 11179 compliant name and definition for each element identified in step 2 that will be used in an information exchange scenario.

Step 5. Identify extra metadata required to understand the business value of each element. This extra metadata may be expressed in either the schema or the instance as attributes (section 7.4 Attributes versus Elements provides detailed guidance).

Step 6. Analyze the information element. Ensure you have identified specific physical elements for each data item that will appear in the XML instance. This process will help the team identify underlying logical elements or generic physical elements that can be reused by declaring them as XML Schema data types or as abstract elements. This analysis should supplement the model you defined in step 2, and may require that you iterate through step 2 again. The UML static structure artifact is extremely useful here. Last, determine relationships between elements defined here and existing data models and definitions (such as the ebXML core components, the DDDS, the DoD XML Registry and Data Emporium).

Step 7. Identify any common business terms that are associated with the information elements defined in step 2. If any are identified, one or more of these will be used as the actual XML element names.

Step 8. Create the schema [6].

a. If creating schema as a DTDs, your choices are to make the model elements just defined an XML element or an attribute

b. If employing the XML Schema language, you have some extra choices in deciding how to express a model element. Model elements can be expressed:

- As types, which may be declared abstract.
- As abstract XML elements.
- As (non-abstract) XML elements or attributes.

One strategy for creating XML Schemas is as follows:

- Create an underlying set of simple and complex XML Schema data types describing base data types, reusable logical and generic physical elements.
- Declare every model element that will appear in the XML instance as type that derives from the types declared previously.
- Create XML Schema data types and attributes using the same name as the ISO 11179 named model elements
- Create XML elements names according to business terms, actor and role names. For instance <***TransmitterUnit***> is a tag name consisting of a role name and an actor name. <***AcousticFrequency***> is a business term for '***Acoustic Signal. Frequency. Measure***'. When no business term, or actor/role exists, consider creating element names that

[6] Up until now, we have not considered how we will express the information in XML. It is a good XML engineering practice to go through the process of defining and modeling information before the additional complications and design alternatives of XML are addressed. Trying to do both information modeling and XML design at the same time is confusing, and often, critical aspects of one or the other are missed.

consist of an optional context term plus the ISO 11179 Object Class (plus property term if appropriate) plus representation term. For example <***DoDMaterialItemIdentifier***>, where the context term is "DoD" indicating that the element is specific to the Department of Defense.

- For business terms with commonly used synonyms, such as NSN for National Stock Number, create a substitution group for the additional synonyms.

c. Build the schema from the bottom-up and top-down.

Step 9. Register any newly created XML elements with the DoD XML Registry.

Appendix C - Tools and References

Tools

Tools for developing and employing XML in applications are flooding the market. However, most if not all of these tools are in early stages of development. In future revisions to this publication, recommendations will be provided as to tools that have either been used, evaluated or are known by reputation. Pros and cons of each will be presented in the case where they are known. Application developers that have used a particular tool may request that it be included in this list, provided it meets at least two of the following criteria:

- It is relatively mature or produced by an established vendor (such as IBM or Microsoft). A beta tool from Microsoft, or from IBM Alphaworks may be included; however, a beta tool from CrazyXMLTools.com should not.
- It is a leader in a developing area, such as X2X's XLink processor. While still immature, it is currently one of the leaders in XLink processing software.
- It has been used by a Navy activity and found to be useful and relatively free of bugs, or the bugs are well documented.
- It has been evaluated by a neutral third party (such as Forrester or the Gartner Group, or an established periodical) with favorable results.

Submit proposed tools to the editor using the format of the following table:

Name & Link	*Description*	*Pros*	*Cons*
XML, XSL and Schema Development			
XML Parsers and XSL Processors			
Databases			
"Servers"			
Miscellaneous			

A more complete list of available XML software is maintained at www.xmlsoftware.com.

Publications

The following publications have been reviewed by the editor and found to be good reference material. The table presents several levels of readers and recommends appropriate reading for each.

Audience	*Title*	*ISBN*	*Author(s)*	*Date*
Management /Business	XML: A Manger's Guide	0-201-43335-4	Dick	2000
	ebXML: The New Global Standard for Doing Business on the Internet	0-735-71117-8	Kotok & Weber	2001
Business / Technical	XML in a Nutshell : A Desktop Quick Reference (Nutshell Handbook)	0-596-00058-8	Harold & Means	2001
	Metadata Solutions: Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand	0-201-71976-2	Tannenbaum	2001
	Modeling XML Applications with UML: Practical e-Business Applications	0-201-70915-5	Carlson	2001
Technical	The Wrox Professional XML Series		Wrox	
	Building B2B Applications with XML: A Resource Guide	0-471-40401-2	Fitzgerald	2001
	Java & XML, 2nd Edition: Solutions to Real-World Problems	0-596-00197-5	McLaughlin	2001
	SOAP: Cross Platform Internet Development Using XML	0-130-90763-4	Seely & Sharkey	2001
	Inside XSLT	0-735-71136-4	Holzner	2001
	XML Schema Development: An Object-Oriented	0-672-32059-2	Brauer	2001

Internet

BizTalk http://www.biztalk.org/home/default.asp

DoD XML Registry: http://diides.ncr.disa.mil/xmlreg/user/index.cfm

ebXML http://www.ebxml.org

eBTWG http://www.ebtwg.org/

OASIS http://www.oasis-open.org/

Open Applications Group http://www.openapplications.org/

The Object Management Group www.omg.org

RosettaNet http://www.rosettanet.org/rosettanet/Rooms/DisplayPages/LayoutInitial

Schema.net http://www.schema.net

W3C http://www.w3.org

XML.com http://www.xml.com/

The XML Cover Pages http://www.oasis-open.org/cover/sgml-xml.html

XML Software.com http://www.xmlsoftware.com/

Appendix D – W3C XML Recommendations

Appendix deleted. A current list may be found at the W3C Technical Reports[xxii] page.

Appendix E – Combined XML Schema Example

The following XML Schema is a combined example illustrating some of the guidance and concepts discussed in this document. The example is non-normative, and does not represent the consensus of the DON XML WG. It is provided for information only.

In this example, a tag from the DoD XML Registry, <***ACOUST_SIGNA_FREQ***> is reused, but the principles of ISO 11179 and camel case are applied using the functionality of the XML Schema language to maintain interoperability.

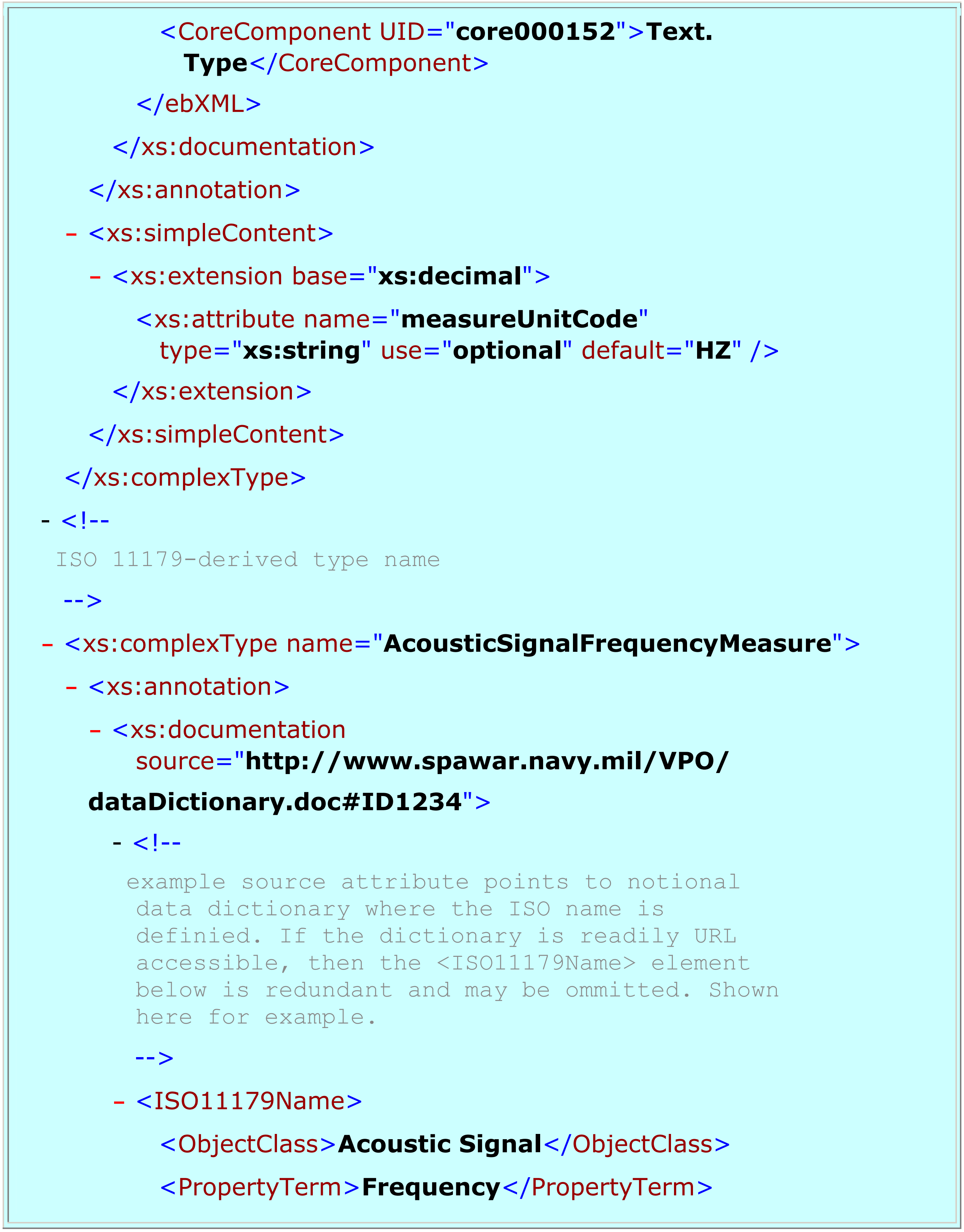
The DoD XML Registry defines a tag <***ACOUST_SIGNA_FREQ***> in the Tracks & Reports Namespace. An instance might look like this:

```
<ACOUST_SIGNA_FREQ>12.100</ACOUST_SIGNA_FREQ>

Definition: ACOUSTIC SIGNATURE FREQ. THE FREQUENCY OF AN
EMITTED ACOUSTIC SIGNAL TO THE NEAREST ONE THOUSANDTH HERTZ.

Maximum Length: 10
```

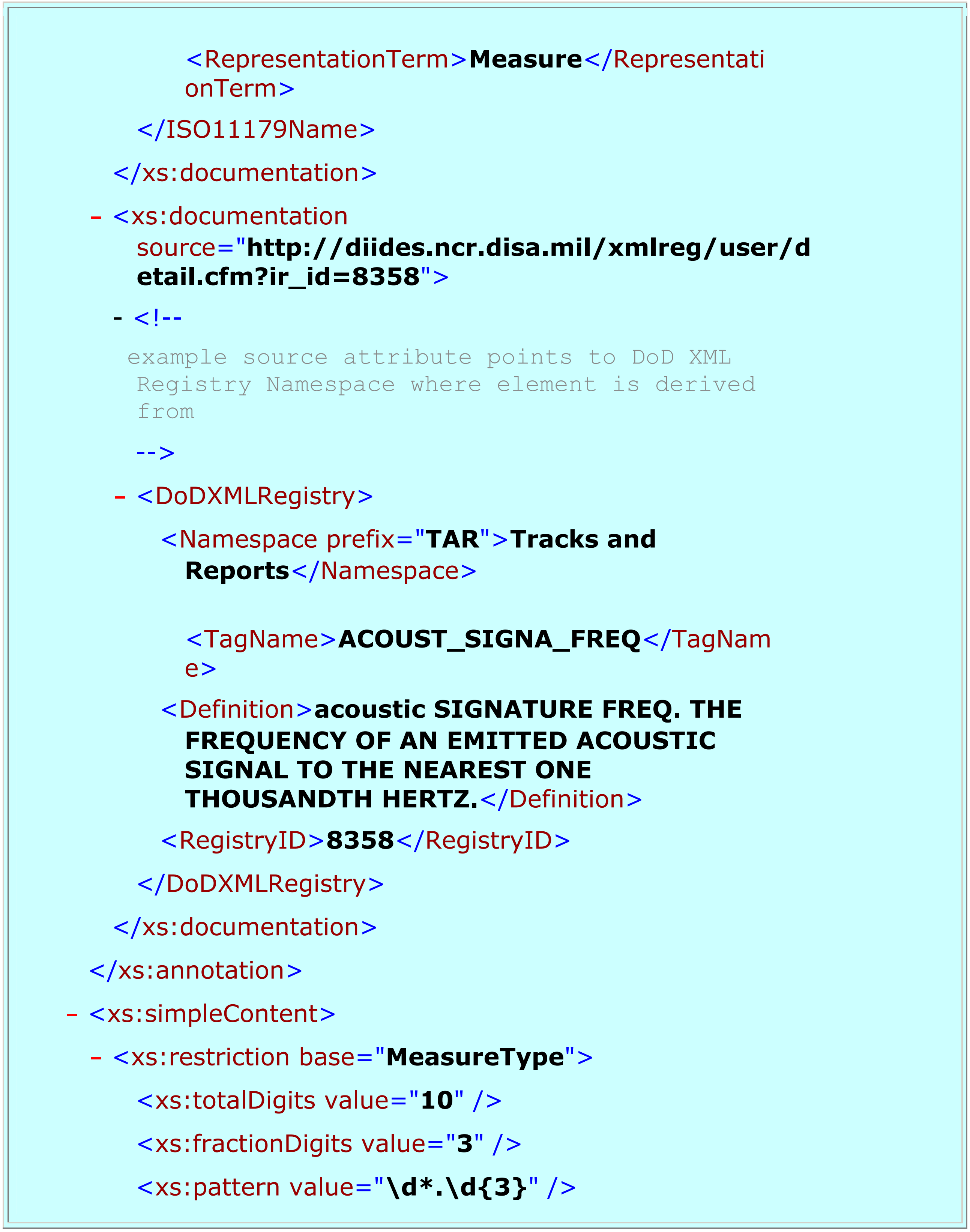
You can view this tag definition at http://diides.ncr.disa.mil/xmlreg/user/detail.cfm?ir_id=8358.

A possible XML Schema for this element:

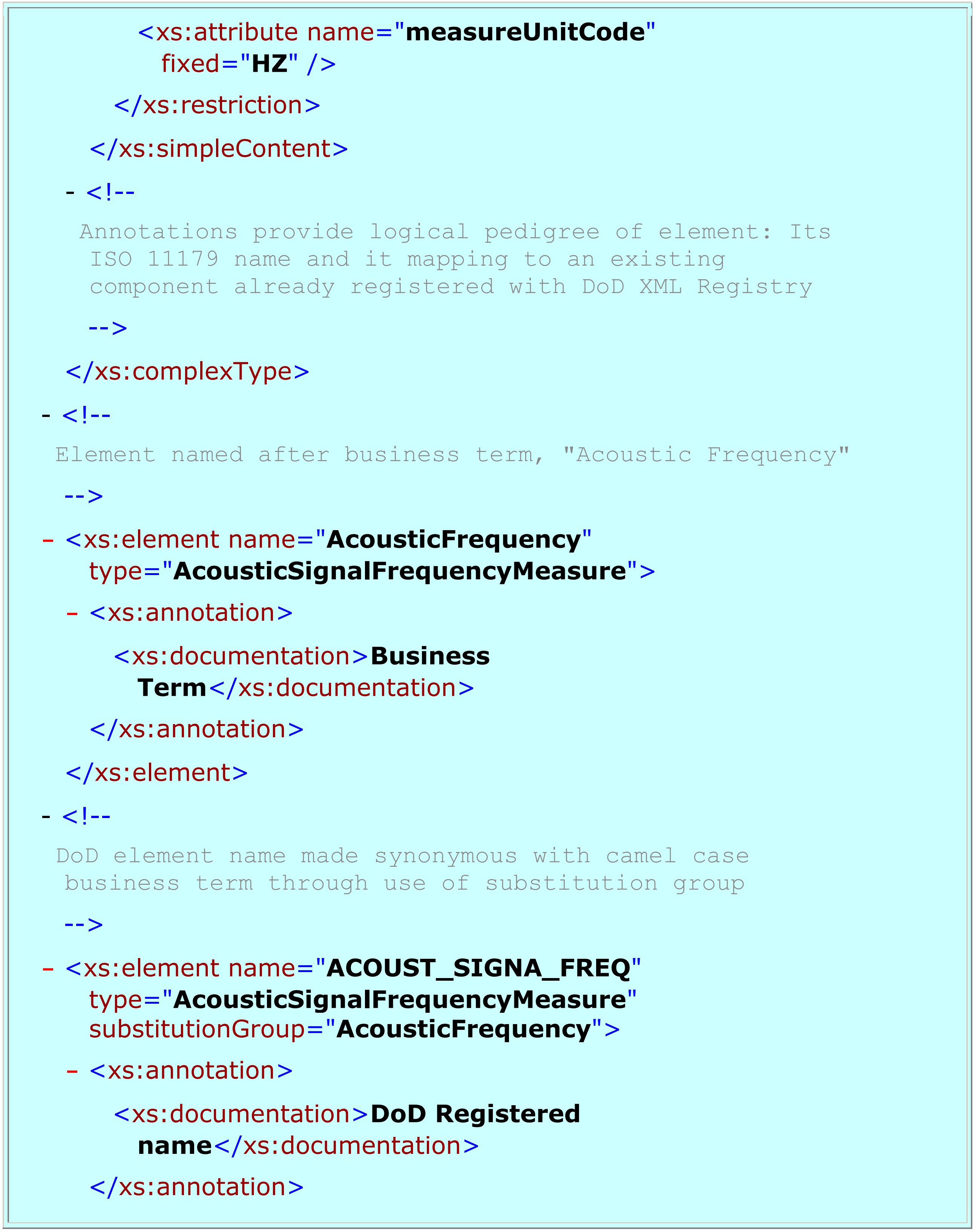
```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 edited with XML Spy v4.1 U
  (http://www.xmlspy.com) by Brian
  Hopkins(Logicon/CISD)
  -->
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  - <xs:complexType name="MeasureType">
    - <xs:annotation>
      - <xs:documentation
          source="http://www.ebxml.org/specs/ccDICT.pdf
          ">
        - <ebXML>
```



```
              <CoreComponent UID="core000152">Text.
                Type</CoreComponent>
            </ebXML>
          </xs:documentation>
        </xs:annotation>
      - <xs:simpleContent>
        - <xs:extension base="xs:decimal">
            <xs:attribute name="measureUnitCode"
              type="xs:string" use="optional" default="HZ" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    - <!--
     ISO 11179-derived type name
      -->
    - <xs:complexType name="AcousticSignalFrequencyMeasure">
      - <xs:annotation>
        - <xs:documentation
            source="http://www.spawar.navy.mil/VPO/
        dataDictionary.doc#ID1234">
          - <!--
           example source attribute points to notional
            data dictionary where the ISO name is
            definied. If the dictionary is readily URL
            accessible, then the <ISO11179Name> element
            below is redundant and may be ommitted. Shown
            here for example.
            -->
          - <ISO11179Name>
              <ObjectClass>Acoustic Signal</ObjectClass>
              <PropertyTerm>Frequency</PropertyTerm>
```



```
          <RepresentationTerm>Measure</RepresentationTerm>
        </ISO11179Name>
      </xs:documentation>
    - <xs:documentation
        source="http://diides.ncr.disa.mil/xmlreg/user/detail.cfm?ir_id=8358">
      - <!--
       example source attribute points to DoD XML
        Registry Namespace where element is derived
        from
        -->
      - <DoDXMLRegistry>
          <Namespace prefix="TAR">Tracks and
            Reports</Namespace>

            <TagName>ACOUST_SIGNA_FREQ</TagName>
          <Definition>acoustic SIGNATURE FREQ. THE
            FREQUENCY OF AN EMITTED ACOUSTIC
            SIGNAL TO THE NEAREST ONE
            THOUSANDTH HERTZ.</Definition>
          <RegistryID>8358</RegistryID>
        </DoDXMLRegistry>
      </xs:documentation>
    </xs:annotation>
  - <xs:simpleContent>
    - <xs:restriction base="MeasureType">
        <xs:totalDigits value="10" />
        <xs:fractionDigits value="3" />
        <xs:pattern value="\d*.\d{3}" />
```



```
          <xs:attribute name="measureUnitCode"
            fixed="HZ" />
        </xs:restriction>
      </xs:simpleContent>
    - <!--
     Annotations provide logical pedigree of element: Its
      ISO 11179 name and it mapping to an existing
      component already registered with DoD XML Registry
      -->
    </xs:complexType>
  - <!--
   Element named after business term, "Acoustic Frequency"
    -->
  - <xs:element name="AcousticFrequency"
      type="AcousticSignalFrequencyMeasure">
    - <xs:annotation>
        <xs:documentation>Business
          Term</xs:documentation>
      </xs:annotation>
    </xs:element>
  - <!--
   DoD element name made synonymous with camel case
    business term through use of substitution group
    -->
  - <xs:element name="ACOUST_SIGNA_FREQ"
      type="AcousticSignalFrequencyMeasure"
      substitutionGroup="AcousticFrequency">
    - <xs:annotation>
        <xs:documentation>DoD Registered
          name</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:schema>
```

Schema Guide for AccousticSignalFrequencyMeasure Schema Type and Associated Elements

The Schema defines 5 XML Components: 2 types, 2 elements and 1 attribute.

Elements	Complex types
ACOUST_SIGNA_FREQ	AcousticSignalFrequencyMeasure
AcousticFrequency	MeasureType

The DoD Registered element name is defined as:

element **ACOUST_SIGNA_FREQ**

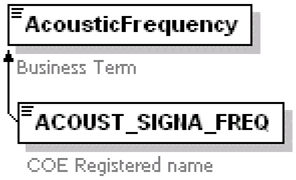
diagram	
type	AcousticSignalFrequencyMeasure
facets	totalDigits 10 fractionDigits 3 pattern \d*.\d{3}
attributes	Name / Type / Use / Default / Fixed measureUnitCode / / / / HZ
annotation	documentation DoD Registered name

source	<xs:element name="ACOUST_SIGNA_FREQ" type="AcousticSignalFrequencyMeasure" substitutionGroup="AcousticFrequency"> <xs:annotation> <xs:documentation>DoD Registered name</xs:documentation> </xs:annotation> </xs:element>

Points to note:

- It is derived from a type '***AcousticsSignalFrequencyMeasure***'
- It has several facets that restrict its domain
- It has one attribute, '***measureUnitCode***' that is fixed with a value of HZ.
- It is declared to be in the substitution group of the element '***AcousticFrequency***'.

element **AcousticFrequency** is a business term (notionally agreed to by all stakeholders within a COI).

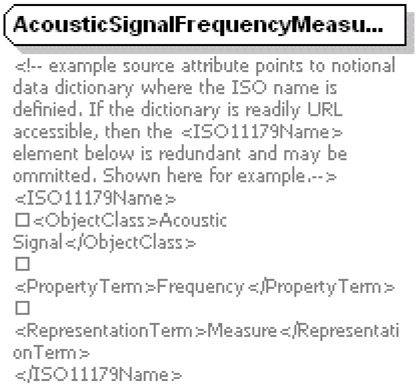
<table>
<tr><td>diagram</td><td>
</td></tr>
<tr><td>type</td><td>AcousticSignalFrequencyMeasure</td></tr>
<tr><td>facets</td><td><table><tr><td>totalDigits</td><td>10</td></tr><tr><td>fractionDigits</td><td>3</td></tr><tr><td>pattern</td><td>\d*.\d{3}</td></tr></table></td></tr>
<tr><td>attributes</td><td><table><tr><td>Name</td><td>Type</td><td>Use</td><td>Default</td><td>Fixed</td></tr><tr><td>measureUnitCode</td><td></td><td></td><td></td><td>HZ</td></tr></table></td></tr>
<tr><td>annotation</td><td><table><tr><td>documentation</td><td>Business Term</td></tr></table></td></tr>
<tr><td>source</td><td>&lt;xs:element name="AcousticFrequency" type="AcousticSignalFrequencyMeasure"&gt;</td></tr>
</table>

	<xs:annotation> <xs:documentation>Business Term</xs:documentation> </xs:annotation> </xs:element>

Points to note:

- The Business Term has a synonym, '***ACOUST_SIGNA_FREQ***', defined above and declared to be in the substitution group.
- It has the same attributes and facets as '***ACOUST_SIGNA_FREQ***' because it derives from the same type.

complexType **AcousticSignalFrequencyMeasure** is the common Schema type from which both elements are derived.

<table>
<tr><td>diagram</td><td>
&lt;!-- example source attribute points to notional data dictionary where the ISO name is definied. If the dictionary is readily URL accessible, then the &lt;ISO11179Name&gt; element below is redundant and may be ommitted. Shown here for example.--&gt;
&lt;ISO11179Name&gt;
□&lt;ObjectClass&gt;Acoustic Signal&lt;/ObjectClass&gt;
□
&lt;PropertyTerm&gt;Frequency&lt;/PropertyTerm&gt;
□
&lt;RepresentationTerm&gt;Measure&lt;/RepresentationTerm&gt;
&lt;/ISO11179Name&gt;</td></tr>
<tr><td>type</td><td>restriction of MeasureType</td></tr>
<tr><td>used by</td><td><table><tr><td>elements</td><td>ACOUST_SIGNA_FREQ
AcousticFrequency</td></tr></table></td></tr>
<tr><td>facets</td><td><table><tr><td>totalDigits</td><td>10</td></tr><tr><td>fractionDigits</td><td>3</td></tr><tr><td>pattern</td><td>\d*.\d{3}</td></tr></table></td></tr>
<tr><td>attributes</td><td><table><tr><td>Name</td><td>Type</td><td>Use</td><td>Default</td><td>Fixed</td></tr><tr><td>measureUnitCode</td><td>xs:string</td><td>optional</td><td></td><td>HZ</td></tr></table></td></tr>
</table>

annotation

documentation
documentation

```
<!-- example source attribute points to notional data dictionary where the ISO name is definied. If the dictionary is readily URL accessible, then the <ISO11179Name> element below is redundant and may be ommitted. Shown here for example.-->

<ISO11179Name>

        <ObjectClass>Acoustic Signal</ObjectClass>

        <PropertyTerm>Frequency</PropertyTerm>

        <RepresentationTerm>Measure</RepresentationTerm>

</ISO11179Name><!-- example source attribute points to DoD XML Registry Namespace where element is derived from -->

<DoDXMLRegistry>

        <Namespace prefix="TAR">Tracks and Reports</Namespace>

        <TagName>ACOUST_SIGNA_FREQ</TagName>

        <Definition>acoustic SIGNATURE FREQ. THE FREQUENCY OF AN EMITTED ACOUSTIC SIGNAL TO THE NEAREST ONE THOUSANDTH HERTZ.</Definition>

        <RegistryID>8358</RegistryID>

</DoDXMLRegistry>
```

source

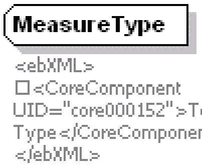
```
<xs:complexType name="AcousticSignalFrequencyMeasure">
  <xs:annotation>
    <xs:documentation source="http://www.spawar.navy.mil/VPO/dataDictionary.doc#ID1234 ">
      <!-- example source attribute points to notional data dictionary where the ISO name is definied. If the dictionary is readily URL accessible, then the <ISO11179Name> element below is redundant and may be ommitted. Shown here for example.-->
      <ISO11179Name>
        <ObjectClass>Acoustic Signal</ObjectClass>
        <PropertyTerm>Frequency</PropertyTerm>
        <RepresentationTerm>Measure</RepresentationTerm>
      </ISO11179Name>
    </xs:documentation>
    <xs:documentation source="http://diides.ncr.disa.mil/xmlreg/user/detail.cfm?ir_id=8358">
      <!-- example source attribute points to DoD XML Registry Namespace where element is derived from -->
      <DoDXMLRegistry>
        <Namespace prefix="TAR">Tracks and Reports</Namespace>
        <TagName>ACOUST_SIGNA_FREQ</TagName>
        <Definition>acoustic SIGNATURE FREQ. THE FREQUENCY OF AN EMITTED ACOUSTIC SIGNAL TO THE NEAREST ONE THOUSANDTH HERTZ.</Definition>
        <RegistryID>8358</RegistryID>
      </DoDXMLRegistry>
    </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:restriction base="MeasureType">
```

```
            <xs:totalDigits value="10"/>
            <xs:fractionDigits value="3"/>
            <xs:pattern value="\d*.\d{3}"/>
            <xs:attribute name="measureUnitCode" fixed="HZ"/>
          </xs:restriction>
        </xs:simpleContent>
        <!-- Annotations provide logical pedigree of element: Its ISO 11179 name and it mapping to an existing
      component already registered with DoD XML Registry -->
      </xs:complexType>
```

Points to note:

- The Type annotation provides
  - ISO 11179 name parts. The source of this documentation is provided as a notional data dictionary referenced by URL and ID.
  - DoD Registry Metadata including the definition
- The domain restrictions are placed in the type vice at the element level.
- The attribute, ‘***measureUnitCode’*** has an optional value of HZ. It is set to fixed in the element declaration.
- The type is derived from an ebXML “core component”

complexType **MeasureType** is a complex type derived from an ebXML core component.

<table>
<tr><td>diagram</td><td colspan="5">

MeasureType
<ebXML>
□<CoreComponent UID="core000152">Text. Type</CoreComponent>
</ebXML>

</td></tr>
<tr><td>type</td><td colspan="5">extension of **xs:decimal**</td></tr>
<tr><td>used by</td><td>complexType</td><td colspan="4">AcousticSignalFrequencyMeasure</td></tr>
<tr><td rowspan="2">attributes</td><td>Name</td><td>Type</td><td>Use</td><td>Default</td><td>Fixed</td></tr>
<tr><td>measureUnitCode</td><td>xs:string</td><td>optional</td><td>HZ</td><td></td></tr>
<tr><td>annotation</td><td>documentation</td><td colspan="4">&lt;ebXML&gt;</td></tr>
</table>

	<CoreComponent UID="core000152">Text. Type</CoreComponent> </ebXML>
source	<xs:complexType name="MeasureType"> <xs:annotation> <xs:documentation source="http://www.ebxml.org/specs/ccDICT.pdf"> <ebXML> <CoreComponent UID="core000152">Text. Type</CoreComponent> </ebXML> </xs:documentation> </xs:annotation> <xs:simpleContent> <xs:extension base="xs:decimal"> <xs:attribute name="measureUnitCode" type="xs:string" use="optional" default="HZ"/> </xs:extension> </xs:simpleContent> </xs:complexType>

Points to note:

- The measureUnitCode attribute common to all other types and elements is defined only once, here.
- The type extends from the simpleType of decimal, again, defined only once here
- The annotations provide mapping to the initial ebXML core component UID.

XML Schema documentation generated with **XML Spy** Schema Editor **www.xmlspy.com**

Some examples of XML instance fragments this document will validate:

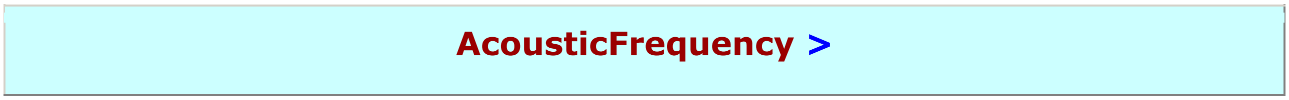
<ACOUST_SIGNA_FREQ>100.000**</ACOUST_SIGNA_FREQ>**

or

<ACOUST_SIGNA_FREQ measureUnitCode="HZ">100.000**</ACOUST_SIGNA_FREQ>**

or

<AcousticFrequency measureUnitCode="HZ">100.000**</**



AcousticFrequency >

Appendix F – Sample XML Document Headers

Sample Schema Header

<?xml version="1.0" encoding="UTF-8">

<!— Schema/DTD Header ***************************

Schema Name: SPAWARVPO$2-1_FolderData$1-1.xsd

DoD Namespace(s): TBD

Navy Functional Data Area: Administration

Current version available at (URL): https://www.spawar.navy.mil/vpo/schemas/SPAWARVPO$2-1_FolderData$1-1.xsd

Other Schemas Imported (XML Schema only):

**** Namespace Prefix: PER "http://diides.ncr.disa.mil/xmlreg/user/namespace_list.cfm"

**** Schema File Name: BUPERSBUPERSOnLine$3-0_Document$2-2.xsd

**** Available at URL: www.bupers.navy.mil/bupersOnLine/schemas/

Other Schemas Included (XML Schema only): None

External DTDs Referenced (DTD only): n/a

**** Name: n/a

**** Available at (URL): n/a

Description: Provides information regarding the content of VPO folders such as content file names, file sizes, file owner, file status, and file access information.

Application: Virtual Program Office

Application Version: 2.1

Application Interface:

XML data is available from the VPO application via HTTP at https://www.spawar.navy.mil/vpo/GetFolderInfo.asp. Input queries via HTTP GET with query string format, "...?dir=directoryName". A complete interface description document is available at https://www.spawar.navy.mil/vpo/interfaces/GetFolderInfo.txt

Associated Stylesheet:

**** Name: SPAWARVPO$2-1_ViewFolderContents$1-0.xsl

**** Available at (URL): https://www.spawar.navy.mil/vpo/stylesheets/

Developed by (Gov't Activity): SPAWAR 08

```
Point of Contact Name: Joe Smith
Point of Contact Email: jsmith@spawar.navy.mil
Change History:
CHANGE #  Version       DATE          DESCRIPTION OF CHANGE
 0           1.0            15 Sep 2001    Initial release
 1           1.1            30 Sep 2001    Updated to include file size information
*********************************************
-->
```

This is a generic header that is provided in text-only, non-XML format. It can be used for either a DTD or XML Schema. A possibly more useful approach would be to markup header information using XML. The tags could be encapsulated by XML comment markup (<!-- ... --> or in the case of XML Schemas, included as an annotation following the XML Schema declaration. Marking up header information could be very useful; for instance, a large number of schemas could be analyzed automatically to determine which DoD Namespaces and Functional Data areas they fell into. This would be a time consuming manual process otherwise. The DON XML WG may work to standardize the tags and procedures for providing header information in XML markup. Until then, it is important to get the information somewhere in the document. Activities wishing to experiment with different strategies and techniques for providing header data are encouraged to do so and report there findings to the DON XML WG. Consider the above example the minimum information we think will be required; your input is encouraged.

Notes on header fields:

Header Item	Description
Schema Name:	The standard name of the schema file. See Document Naming Convention
Tested With:	List the name and version number of the XML processor(s) that have been are tested known to corectly validate this schema.
DoD Namespace(s):	Identify the DoD Namespace that the elements from this schema are registered in by specifying the DoD XML Namespace Prefix from the DoD XML Registry. You can specify muliple Namespaces for XML Schemas that use tags from mulitple DoD Namespaces. This is only possible through the use of XML Schemas because DTD's do not support XML Namespace prefixing.
Functional Data Area:	Indicate which Navy Functional Data Area the application that uses this schema belongs to. Refer to the DMI Instruction (SECNAVINST 5000.36) and implementation guidance for a list.
Current version available at (URL):	If this schema is URL accessable, put the address here. It is highly recommended that all schemas be available on-line to assist other activities desiring interoperabiity

activities desiring interoperabiity.

Other Schemas Imported (XML Schema only):

The next three fields are repeatable

The XML Schema language allows the reuse of existing XML Schema so that schemas can be modularized. The first way of doing this is via the XML Schema Import syntax.

***** Namespace Prefix and URL:*

The XML Schema Import syntax is used when desiring to reuse a schema whose elements belong to a different XML Namespace than the elements into which the import is being conducted on. Specify here

***** Schema File Name:*

The standard name of the imported schema file. See Document Naming Convention

**** Available at (URL):*

If this schema is URL accessible, put the address here. It is highly recommended that all schemas be available on-line to assist other activities desiring interoperability.

Other Schemas Included (XML Schema only):

The next two fields are repeatable

The second way XML Schemas allow reuse of other schemas is through the XML Schema Include syntax. Includes can be used when the elements in the included schema belong to the same XML Namespace as the schema into which the include is occuring. A schema may both include and import.

***** Schema File Name:*

The standard name of the imported schema file; see Document Naming Convention

**** Available at (URL):*

If the schema file to be imported is URL accessible, put its address here. It is highly recommended that all schemas be available on-line to assist other activities desiring interoperability.

External DTDs Referenced (DTD only):

The next two fields are repeatable

Information regarding any External Parameter Entity references are made to an external DTD. This approximates the modular design capability available in XML Schema.

***** Name:*

The standard name of the DTD file; see Document Naming Convention

***** Available at(URL):*

If this schema DTD is URL accessible, put its address here. It is highly recommended that all schema DTDs be available on-line to assist other activities desiring interoperability.

Description:

Plain text description of the type of information described by the schema.

Application:

The name of the application which produces XML documents that validate to this schema.

Application Version:

The version (major.minor) of the application that produces this schema.

Application Interface:

A plain text descriptive summary of how other applications interface with this application. For example, via HTTP, using query parameters passed via HTTP POST or GET. Examples of query name/value pairs may be provided. If SOAP is used, should provide a brief description of the method calls and parameters. A good XML engineering practice is to completely document your application interface; if you have done so, reference that documentation here. Making the interface specification available via a (secure) URL will assist other developers in interoperating.

Associated Stylesheet:

If a stylesheet is available to render instances that validate to this schema, provide information here.

***** Name:*

The standard name of the stylesheet file; see Document Naming Convention

***** Available at (URL)*

If the stylesheet is URL accessible, put the its address here. It is highly recommended that all stylesheets be available on-line to assist other

	activities desiring interoperability.
Developed by (Gov't Activity):	Government Activity and Office code.
Point of Contact Name: Joe Smith	Name of person to contact with questions regarding the schema.
Change History:	The following fields provide an audit trail of changes.
CHANGE #	Keep a sequentially numbered list of changes.
Version	You should also assign Major and minor version numbers.
DATE	Date implemented
DESCRIPTION OF CHANGE	Plain text description.

Sample Stylesheet Header

This sample stylesheet header is the similar to the schema header with the addition of information regarding which version of a schema the stylesheet is written from, and the removal of non-applicable items.

```
    <?xml version=”1.0”>
    <!— Stylesheet Header ***************************
    Stylesheet Name: SPAWARVPO$2-1_ViewFolderData$1-1.xsl
    Tested to:
    **** Schema Name:  SPAWARVPO$2-1_FolderData$1-1.xsd
    **** Schema Version:  1.1
    **** XSL Processors: MSXML 3.0, XALAN 1.2.2
    DoD Namespace: TBD
    Navy Functional Data Area: Administration
Current version available at (URL): https://www.spawar.navy.mil/vpo/stylsheets/
    Other Stylesheets Imported:
    **** File Name: BUPERSBUPERSOnLine$3_Document$2-2.xsl
    **** Available at URL: www.bupers.navy.mil/bupersOnLine/stylsheets/
    Description: XSLT compliant stylesheet renders folder contents as an HTML table
    Application: Virtual Program Office
    Application Version: 2.1
    Developed by (Gov’t Activity): SPAWAR 08
    Point of Contact Name: Joe Smith
    Point of Contact Email: jsmith@spawar.navy.mil
    Change History:
    CHANGE #  Version       DATE        DESCRIPTION OF CHANGE
     0          1.0          15 Sep 2001   Initial release
     1          1.1          30 Sep 2001   Updated to include file size information
    *********************************************
    -->
```

The following notes indicate differences between the stylesheet and schema header only.

Header Item	Description
Stylesheet Name:	The standard name of the schemastylesheet file. See Document Naming Convention
Tested to:	Information regarding the specific schema and software this stylesheet has been tested with.
`**** Schema Name:`	Name(s) of the schemas this stylesheet has been tested with.
`**** Schema Version:`	Version(s) of the schemas this stylesheet has been tested with.
`**** XSL Processors:`	Name(s) of the XSL processors this stylesheet has been tested with.
Other Stylesheets Imported	Stylesheets, like schema, can be constructed modularly. Provide information here regarding other stylesheets reused.
The next two fields are repeatable	
***** File Name:*	
**** Available at (URL):*	If this Stylesheet is URL accessible, put its address here.

Sample Instance header

It is important that XML documents include some basic information. Most of the needed information can be gleaned from the header data provided by the schema that describes the document and the stylesheet(s) that transform or render it. The XML specifications provide syntax for pointing to schemas and stylesheets at the beginning of an XML document. In cases where validation against a schema and/or transformation with a stylesheet is not required, it is still desirable to provide references to schemas and stylesheets if available. Consider this example:

```
<?xml version="1.0" encoding="UTF-8" ?>
    <! --
    Schema and Stylesheet Reference Data:
    stylesheet type = xslt
               url =
    http://spawar.navy.mil/stylesheets/SPAWARVPO$2-
    1_ViewFolderData$1-1.xsl
               version = 1.1
    schema type = XML Schema (W3C)
           url = http://spawar.navy.mil/schemas/SPAWARVPOV2-
    1FolderDataV1-1.xsd
           version = 1.1
  -->
  <root />
```

Appendix G – Draft Glossary and Acronyms

The following draft glossary is provided in advance of the DON XML WG's future XML Glossary deliverable. It represents the understanding and opinion of the editor, and does not reflect the consensus of the DON XML WG. These items are provided for information only.

Some terms may have "(XML)" prepended. This convention indicates that the term has meaning other than in the context of XML, and that the definition applies only to the XML context.

Terms

Abstract – In the context of an XML Schema, an XML element or Schema type may be declared abstract, meaning that it may not be used directly. An abstract element may not be used directly in an instance, but must have in its substitution group a non-abstract element. For instance, an abstract element, 'Address', defines the contents of an address. A non-abstract 'HomeAddress' element that is substitutable for 'Address' can be used as an XML element. The 'HomeAddress' structure reuses the previously defined 'Address' contents, but the tag provides a specific context. Schema types may also be declared abstract. Similar to abstract elements, abstract types may not be directly used to reference elements, but must have a non-abstract type that extends/restricts it. The non-abstract type can then be used to reference XML elements. The concept of abstractness is taken from object-oriented programming, where an abstract class may be defined, requiring subtyping prior to instantiation.

Binding - A term frequently used in reference to XML applications taken from the field of computer science. In the context of applications that have a public interface that communicates in XML (such as the case with a web service), binding refers to the information required and the process by which an external source connects to, and interacts with it to get data in XML. Binding can also refer to the process and application required to connect a software module (e.g. a Java class, or COM object) to a public XML interface or the way in which the public XML is related to an underlying data source (such as a relational database).

BPSS - The Business Process Specification Schema was developed as part of the ebXML project as a schema for describing a business process in an XML instance. It may be created from UML models of business processes developed according to the UMM as described in the technical report, Business Process and Business Information Analysis Overview v1.0[xxiii]. The BPSS is available in either DTD format [xxiv] or XML Schema (Candidate Recommendation) format[xxv].

Business Term - The ebXML specifications refers to a *business term* as a commonly used term referencing a commonly understood concept within a specific domain. To enhance understanding, it is appropriate to use business terms as XML Element names (when they exist), rather than the often esoteric ISO 11179 syntax.

C4ISR – Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance

Camel Case – A convention in which names of elements and attributes are all lower case with the exception of the beginning of a new word, which is in uppercase. ebXML differentiates between *upper* camel case where the first letter of the name is also capitalized and *lower* camel case where it is not. Example of an upper camel case name: UpperCamelCase. A lower or just camel case name: lowerCamelCase. Camel case is emerging as the industry norm for XML element naming. ebXML specifies elements to be in upper and attributes to be in lower camel case, while BizTalk, RosettaNet, and Oasis use straight camel case for both elements and attributes.

CSS - Cascading Style Sheets. A set of W3C recommendations for styling HTML and XML documents based on the application of formatting instructions in a linear, cascading fashion. CSS is an alternative to styling XML with XSL, but CSS does not have the transformational component of XSLT.

Class – A software component that provides instructions for the creation of an object. Applications are said to create *instances* of a class ("objects") through a process referred to as *instantiation*. In the context of XML, a schema is a "class" that describes XML instances (data "objects").

COM Object – The Common Object Model is a Microsoft sponsored interface specification for creating interoperable software components. Distributed COM or DCOM is Microsoft's COM interface standard for distributed computing, i.e., where an "application" consists of software "objects" distributed across nodes of a network. DCOM is similar to the Java based EJB specification, but works only for Microsoft operating systems. DCOM objects can communicate via TCP/IP and their own proprietary messaging framework (Windows Distributed iNternet Architecture or DNA). Alternatively, COM objects can communicate with other non-COM / non-Windows objects such as Java Classes or EJB's via XML and SOAP.

CORBA – Common Object Request Broker Architecture. CORBA is a framework created by the Object Management Group[xxix] (OMG) to facilitate platform / operating system / programming language neutral distributed computing. Software components or "objects" interact in client-server relationships, with an Object Request Broker (ORB) software component acting as intermediary. Via the IIOP, CORBA-based distributed applications can operate across the Internet. CORBA is language independent.

Core Components – One goal of the ebXML effort is to define a set of universal, core components that are contextually neutral and can be used across all domains to express semantics of common business concepts. Core components may be information entities, defined in the ebXML Core Component Dictionary technical reports, or process components discussed in the ebXML Business Process technical reports. Note that the core component technical reports do not address how an information component will be expressed in XML – this was an intentional omission on the part of ebXML. It was felt that prior to defining rules for creation of XML, a necessary first step was to create a schema neutral standard for defining components in business terminology. The work of defining how core components map to XML will be undertaken by the Core Component Project Team[xxx] of the UN/CEFACT sponsored Electronic Business Transition Working Group (eBTWG).

DDDS – The Defense Data Dictionary System[xxxi] defines standard data elements per the DoD 8320 series of documents[xxxii]. The DDDS provides definitions of Standard Data Elements (SDEs) from core data models across all DoD data domains. The DDDS elements are mainly logical in nature, and may be used to express logical, semantic relationships between XML elements. XML Schema data types may be used to express relationships to DDDS standard data elements.

Data-centric – Describes the exchange of information between applications where the data being exchanged is sufficiently well defined and granular for transactional processing. In the context of XML, a data-centric markup strategy provides sufficient metadata for non-ambiguous application processing of received data .

Example:

```
<PartDescription>
  <PartSize measureUnitCode=”inch”>1</PartSize>
  <PartType threadDirectionCode=”left”>Wing Nut</PartType>
  <PartNumber>123456</PartNumber>
</ PartDescription >
```

Compare to the document-centric example containing the same information.

Document-centric – Describes the exchange of information, where the data being exchanged is meant to be read and understood by a human. In the context of XML, describes the use of markup to describe information of a non-transactional nature consisting of string data. The string must be read and understood by a human in order to be useful.

Example: <PartDescription>123456, 1” left threaded wingnut</PartDescription>.

Compare to the data-centric example containing the same information.

Document Type Declaration – A declaration at the beginning of an XML document indicating a DTD to which the instance must conform.

DoD XML Registry – The DoD XML Registry[xxxiii] “...provides a baseline set of XML components developed through coordination and approval among the DoD community. The Registry allows you to browse, search, and retrieve data that satisfy your requirements.” DON XML Policy requires that all activities developing XML in the DON register components developed with the appropriate DoD XML Namespace.

DoD Namespace – The DoD XML Registry is divided into “Namespaces”. ”A Namespace is a collection of people, agencies, activities, and system builders who share an interest in a particular problem domain or practical application. This implies a common worldview as well as common abstractions, common data representations, and common metadata. The COE Data Emporium, including the

XML Registry, allows Namespaces to publish their existence and their available information resources so that outsiders may discover them and assess whether or not they want to share." A DoD XML Namespace is an extension of the XML Namespace concept. The terms "XML Namespace" and "DoD XML Namespace" are not synonymous.

DoD Namespace Manager – Each DoD XML Namespace has a central activity responsible for it. The individual responsible for coordinating and administering the Namespace is the Namespace manager. Point of contact information for the Namespace Managers is available by clicking on the Namespace hyperlinks[xxxiv] on the registries web site.

DoD XML Namespace Prefix – Each DoD XML Namespace has been assigned a three-letter prefix that may be used as XML Namespace qualifiers in XML instances and Schemas.

DoD XML Registration Package – Activities developing XML within the DON are required to submit a specially formatted package of information to the DoD Registry containing metadata about the components registered. Information about how and what to register can be found here[xxxv].

DOM - The Document Object Model. The set of W3C DOM recommendations[xxxvi] form application interface descriptions (APIs) for expressing the contents of XML or HTML "documents" as hierarchical tree-like models of information with data forming the "leaves" of the tree. XML Processors that implement the DOM interface parse an entire XML document, creating a data tree in memory. Applications that call a DOM parser access data from the XML object tree through a set of programmatic instructions defined by the specifications. The instructions allow applications to "walk the document tree", searching for elements and attributes that meet query criteria (XPath expressions). Results are returned to the calling application and assigned to application variables for further processing.

DTD - Document Type Definition. A schema syntax that is part of the XML 1.0 specification and derived from SGML.

EJB – Enterprise Java Beans. EJB is an interface specification which a Java class may implement. Software objects that implement the EJB interface may interoperate in an enterprise (distributed) environment, even across the Internet via TCP/IP and the CORBA IIOP. In this fashion, an "application" may consist of a number of independent software components ("objects") that are physically separated at different nodes of a network, but functioning together as a single application similar to the Microsoft (D)COM specification.

Entity – In the context of a DTD, an entity is a declarative construct referencing text, or a binary file. Entities are defined in the DTD, and referenced elsewhere in the DTD (*parameter entity*) or in the body of the XML (*general entity*). A validating parser encountering a reference to a previously defined entity during the validation process will insert the entity's value in place of the entity reference. *Internal entities* are declared in the DTD and may be general or parameter. External entities point to an external file containing the entity declaration via URI reference; they also may be internal or external. A parsed entity is some form of encoded text and is therefore processed by a parser. An unparsed entity is a reference to a binary file that will not be parsed. Unparsed entities are always external. Through entities, DTDs may declare a common construct once, and reuse it many times throughout the DTD or in the instance. A common use for parameter entities is to declare a common set of attributes in the DTD. Assigning the attributes to an element only requires a reference to the parameter entity, vice retyping the entire attribute list many times. A second use of external unparsed general entities is to make reference to a binary file (such as an image or sound file) within an XML instance.

EDI – Electronic Data Interchange. A term referring to the conduct of eBusiness through the exchange of electronic messages. Two message standards exist as rigorously defined sets and segments, one maintained by the U.S. led ANSI X12 body, and the second led by UN/EDIFACT.

Fatal Error - [From the XML 1.0 specification*] "An error which a conforming XML parser must detect and report to the application. After encountering a fatal error, the parser may continue processing the data to search for further errors and may report such errors to the application. In order to support correction of errors, the processor may make unprocessed data from the document (with intermingled character data and markup) available to the application. Once a fatal error is detected, however, the processor must not continue normal processing (i.e., it must not continue to pass character data and information about the document's logical structure to the application in the normal way)."* In other words, upon detecting a fatal error (such as a well-formedness violation), the parser is unable to provide information from the XML document to the calling application such that the application may continue functioning normally.

Functional Area – DMI (SECNAVINST 5000.36) divides DON data administration responsibilities by into functional areas of responsibility. The concept of a functional area is derived from DoD 8320.1.

HTML - Hypertext Markup Language[xxxvii]

Interface – The process by which a software application interacts with other software or users. In object-oriented programming an (software) “object’s” interface is often described separately from the internal logic in a process know as “encapsulation”. Essentially the interface encapsulates and hides the internal logic. This allows flexibility to change and improve object code without affecting other objects. An interface description is made public so other objects/applications know how to interact. Software is said to “implement” an interface if it conforms to the behavior as defined in an interface description. The Object Management Group (OMG) has defined a formal syntax (language) for defining interfaces in a programming language neutral fashion. This is called the OMG Interface Description Language[xxxviii] (OMG IDL). This IDL is used to define interface specifications such as the DOM API and CORBA. For developers implementing public XML interfaces, it is a good idea to document exactly how other applications connect, query, and receive (i.e. bind to) your application; while it is not necessary to go to the trouble of writing a formal IDL interface description, some kind of formal document will greatly aid other applications desiring to share data.

IIOP – Internet Inter-ORB Protocol. A TCP/IP based protocol that facilitates communication between CORBA ORBs. Via IIOP, CORBA client objects at one location on the Internet can communicate with CORBA server objects at another node and vice versa.

ISO 11179 - *Information Technology - Specification and Standardization of Data Elements* is a 6-part ISO standard providing a framework and methodologies for developing, documenting, and registering standard data elements. Of interest to XML developers is Part 5: *Naming And Identification Principles For Data Elements* upon which the ebXML naming convention is based. The specifications are available from the ISO Store[xxxix] under section 35.040 - Character Sets And Information Coding for a small fee.

Markup - Special characters used by Markup Languages (SGML, XML, HTML) to differentiate data from metadata. SGML allows document authors the flexibility of specifying which characters are used for markup, whereas in XML the markup characters are fixed. Markup characters may not be used in data text (unless special precautions are taken). In the tags definition example, the markup characters are '<' (greater than), '>' (less than), and '/' (forward slash). The XML specification[xl] defines start tag markup as opening with a '<' and ending with a '>'. It specifies end tag markup as opening with a '</' and endings with a '>'.

Metadata - Data about data. For example, for the data '3000N', the metadata might be 'latitude'. Markup languages such as SGML and XML encapsulate data with tags that contain text describing the metadata. See the example provided in the tags definition.

Normative – A term frequently used in software specifications to identify requirements. An implementation that conforms to the specification must satisfy all the normative requirements. Non-normative text is provided for information only. A common example of non-normative text is “rationale.”

Object – A term used frequently in relation to XML and object-oriented programming. Strictly speaking, an object is a run-time software construct that resides in the host computer’s memory space. Objects are created by applications from code that defines the object’s behavior; this code is called a class. In object-oriented programs, objects interact with other objects to create the behavior of the application. An object’s behavior is described by an Interface consisting of *methods* and *properties*. A method can be thought of as a behavior of the object that can be triggered by calling it and optionally passing parameters. For instance, the object ‘***myAccount***’ might have the method ‘***getBalance(accountNumber)***’. Object oriented languages use the ‘dot’ notation to refer to objects and methods. From the previous example, ‘***currentBalance == myAccount.getBalance(accountNumber)***’ is a code snippet that assigns to the ‘***currentBalance***’ variable the balance returned from the ‘***myAccount***’ object when the ‘***getBalance()***’ method is called by passing in the ‘***accountNumber***’ variable. Object properties are similar to methods, but instead of calling a behavior, a property call to an object returns a previously set value of the property. Returning to the example, '***myName == myAccount.accountOwner***' sets the '***myName***' variable equal to the '***accountOwner***' property of the '***myAccount***' object, conversely '***myAccount.accountOwner == myName***' sets the '***accountOwner***' property of the '***myAccount***' object to the value of the '***myName***' variable. XML that has been parsed by an XML processor implementing the DOM API is transformed into a set of objects that may be used by the calling application to extract data from the XML. Also, an application may construct a DOM tree of objects in memory then transmit the data to another application or object as a textually encoded string of XML. The receiving object then accesses the data via the DOM or SAX APIs. Since the XML format is neutral, a COM object created by a Windows application may interact with an EJB object running on a Unix platform via XML for true cross-platform, language-independent distributed computing.

Payload (XML) – Protocols and frameworks such as SOAP, BizTalk, and ebXML use XML to mark up message header information necessary for binding, reliable messaging, and security. The term '*payload'* refers to the XML being transmitted that contains the actual business information communicated.

Public (XML) Interface – XML may be employed internally to an application or it may be used to communicate information to another system outside the originating application's environment. The term '*Public Interface*' refers to XML used by an application or set of homogeneous applications to communicate with other applications across system boundaries. DoD and DON policy for registration of XML components applies to public interfaces; these policies are not intended to restrict the use of XML internal to systems; in fact, it is recommended that applications separate internal XML grammars processed by application code from that used for external communications.

Qualified (elements and attributes) – The practice of prefixing an element or an attribute with an XML Namespace qualifier in accordance with the Namespaces in XML[xli] W3C Recommendation. This allows two elements with the same name to be distinguished by an XML processor.

Regular Expression – A language element for defining patterns in strings and numbers. The XML Schema language allows elements and attributes to be constrained by regular expressions to provide a precise description of the range of possible values. For instance an element of type='integer' could be further constrained to be only a 3 digit integer by the regular expression '/d{3}'.

Rendering (XML) - XML is not easily legible to readers in its native format and should be transformed for presentation (i.e., rendered for presentation), either by a CSS, XSLT (to well-formed HTML) for browser viewing, or by XSL-FO into a format for viewing by another presentation application (e.g. into Adobe Acrobat .pdf, or MS Word .doc files.) Note: It is a common assumption that all XML must be rendered (by a stylesheet) to be useful and therefore all XML must have a stylesheet. This is a mistake; XML data can be used by an application via an API and never get rendered at all.

SAX - Simple API for XML. SAX[xlii] is an open-source interface for accessing information from XML documents. SAX parsers process a document, triggering events in the calling application corresponding to the parser encountering opening tags, closing tags and character data. Accessing XML data via SAX is very quick and places fewer demands on system resources than DOM;, however, once processed, a document must be re-parsed if the required information was not retained initially. This can be conceptualized as “serial” access to the information.

Schema - Within the context of XML, a document describing a set of XML Instances. Schemas may be expressed in a number of different languages. Most familiar is the Document Type Definition (DTD) syntax described in the XML 1.0 specification. Schemas provide the rules against which a validating parser validates an instance of XML.

SGML - The Standard Generalized Markup Language [ISO 8879[xliii]]. SGML is the parent of both HTML and XML.

SOAP - "SOAP is the Simple Object Access Protocol, a way to create widely distributed, complex computing environments that run over the Internet using existing Internet infrastructure. SOAP is about applications communicating directly with each other over the Internet in a very rich way." [MS] “SOAP is a protocol specification for invoking methods on servers, services, components, and objects. SOAP codifies the existing practice of using XML and HTTP as a method invocation mechanism. The SOAP specification mandates a small number of HTTP headers that facilitate firewall/proxy filtering. The SOAP specification also mandates an XML vocabulary that is used for representing method parameters, return values, and exceptions." [DevelopMentor]. [Taken from the XML Cover Pages[xliv]]. The current SOAP 1.1 specification[xlv] is a W3C Note; SOAP 1.2[xlvi] is going through the W3C consensus process[xlvii] and was published as a first working draft in July 2001.

SQL - Structured Query Language - A language for querying, writing to, and constructing relational databases. Many versions of SQL exist, meaning that an SQL query that works for one database will not necessarily work against another.

SDE – Standard Data Element as defined by the DoD 8320 series and used in the DDDS.

Stylesheet - A generic term that may refer to an XSL Stylesheet or a CSS. Often the term used to reference XSL Stylesheets implicitly; however, this is not technically correct as a stylesheet may by CSS conformant, and having nothing the do with XML whatsoever. The primary function of a stylesheet is to render XML to a presentation format. However, XSLT can transform one XML instance into another different instance. Application of a stylesheet by an XSL processor to an XML document for the purpose of creating another XML document (i.e. an XML to XML transformation) does not render a presentation format at all. More simply, applying a stylesheet to XML doesn’t imply that the output is ready for viewing; you have to understand what the stylesheet is doing.

Substitution Group – In the context of XML Schemas, a substitution group may be declared for an element to define a synonymous group of tag names. A top-level element is declared, then other elements are declared with an attribute indicating they belong in the substitution group of the top element. Different elements do not necessarily have to have the same structures – used in this fashion they are functionally similar to a group of optional elements where only one may be chosen. The top-level element may be declared abstract; in this case the top level element may not be used but can serve as a generic model for non-abstract elements in the substitution group. This is similar and somewhat redundant of the functionality provided by XML Schema data types.

Throw (an error) – A terms adopted from the Java language to indicate that a processing error has occurred. Conceptually, Java "throws" the error to an error-handling object, which "catches" it, or may "throw" it to another object, and so on.

UID – Unique Identifier. A generic term used to indicate that an object or item has a string or number that identifies it uniquely within a specific context or environment. Universally Unique Identifiers (UUIDs) and Globally Unique Identifiers (GUIDs) are special identifiers that are guaranteed universal uniqueness via an identifier assignment algorithm.

UML - The Unified Modeling Language[xlviii] defines a standard language and graphical notation for creating models of business and technical systems. UML is not only for programmers, it defines several model types that span a range from functional requirements definition and activity work-flow (business process) models to logical and physical software design and deployment. The UML has over the last few years become the *lingua franca* for business and technical stakeholders to communicate and develop IT systems. Through the UMM, UML has been adopted by UN/CEFACT and ebXML as the modeling language of choice.

UMM - The Unified Modeling Methodology[xlix] is a product of UN/CEFACT, and describes the CEFACT recommended methodology for modeling business processes to support the development of the next generation EDI. It is based upon the Rational Unified Process[l], and uses the UML as its modeling language. In the UMM, business processes are modeled by deconstructing them into a series of document exchanges which are orchestrated to form a complex process. The ebXML Technical Report, Business Process and Business Information Analysis Overview v1.0 ,further develops the UMM. The ebXML Business Process Specification Schema v1.01 (BPSS) provides a schema in the form of a DTD for specifying business processes as an XML instance; it may be developed as part of a UMM modeling process.

URL / URI / URN – Uniform Resource Locators, Uniform Resource Indicators, and Uniform Resource Names are different, related methods of uniformly referencing resources across networked environments. A recently release W3C Note explains the difference[li].

Valid (XML) - An XML instance (document) whose structure has been verified in conformance to a schema by a validating parser. Note that an XML instance must be well-formed to be valid, but it does not need to be valid to be well-formed. This is because a parser will always check well-formedness constraints but will only check validation constraints if it is a validating parser.

Validating Parser - An XML parser that enforces validity constraints by comparing the structure and syntax of an XML instance to the rules specified in a schema. Not all parsers are validating parsers, and validating parsers enforce validation according to specific schema languages. Most validating parsers are capable of enforcing validity against a DTD, while some can enforce validation rules described in other schema languages.

Voluntary Consensus Standards – From OMB Circular A119, " Voluntary consensus standards bodies" are domestic or international organizations which plan, develop, establish, or coordinate voluntary consensus standards using agreed-upon procedures. For purposes of this Circular, 'voluntary, private sector, consensus standards bodies,' as cited in Act, is an equivalent term. The Act and the Circular encourage the participation of federal representatives in these bodies to increase the likelihood that the standards they develop will meet both public and private sector needs. A voluntary consensus standards body is defined by the following attributes:

(i) Openness.

(ii) Balance of interest.

(iii) Due process.

(vi) An appeals process.

(v) Consensus, which is defined as general agreement, but not necessarily unanimity, and includes a process for attempting to resolve objections by interested parties, as long as all comments have been fairly considered, each objector is advised of the disposition of his or her objection(s) and the reasons why, and the consensus body members are given an opportunity to change their votes after reviewing the comments. "

Examples of these types of organizations are the W3C and OASIS.

W3C - The World Wide Web Consortium[lii] was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has more than 500 Member organizations[liii] from around the world and has earned international recognition for its contributions to the growth of the Web.

W3C Recommendation - A work that represents consensus[liv] within W3C and has the Director's stamp of approval. W3C considers that the ideas or technology specified by a Recommendation are appropriate for widespread deployment and promote W3C's mission.

W3C Note – A W3C Note is a publication of a member idea. Notes do not go through the consensus process; they represent the ideas of a single (group of) W3C member(s).

(W3C) XML Schema - A schema written in according the W3C XML Schema language. [From the W3C Schema[lv] page] "XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents. The XML Activity Statement[lvi] explains the W3C's work on this topic in more detail." The W3C XML Schema language is described in three recommendations: *XML Schema Part 0: Primer*[lvii]*, XML Schema Part 1: Structures*[lviii]*, and XML Schema Part 2: Datatypes*[lix]. In the DON XML Developers Guidance (this document), the term *XML Schema* will be used in reference to a W3C XML Schema language compliant schema.

Web-service – A generic term used to refer to the use of Hypertext Transfer Protocol (HTTP) and XML to exchange information. Frequently the term implies the use of SOAP to exchange information between applications, vice application to human, which is done in HTML.

Well-formed (XML) - An XML instance that meets well-formedness constraints defined by the XML 1.0 specification. Well-formedness constraints are precise syntactic rules for markup of data. As an example, the XML specification stipulates every open tag must have a corresponding and properly nested closing tag. A document must be well-formed in order to be considered XML. A parser processing a document will throw a fatal error if it detects a well-formedness violation.

Well-formed HTML - HTML that meets the well-formedness constrains of XML 1.0. Well-formed HTML is not the same as XHTML.

XHTML - Extensible HyperText Markup Language[lx].

XML - [From the XML 1.0 specification] *"Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML. By construction, XML documents are conforming SGML documents."* The XML 1.0 specification is a W3C Recommendation. In XML, metadata is described by an extensible set of tags; the tags are said to be extensible, because unlike HTML, where the markup tags are fixed, developers are given the flexibility to define their own tags or reuse tags defined by another party. This flexibility is both the key to XML's power and the single biggest stumbling point to achieving interoperability when making use of XML.

(XML) API - Application Programming Interface. In the context of XML, parsers expose their data to a calling application via an interface. An interface is a specification (which the parser conforms to) that describes how the parser will pass data from an XML document to a calling application. The two accepted XML API's are DOM and SAX.

(XML) Attributes – In the context of XML, attributes provide a mechanism for attaching additional metadata to an XML element. For example, <element attribute="value"/>. An XML attribute is not equivalent to an object or relational model attribute. Data model entity attributes may be expressed as either XML attributes or elements. Frequently in discussions surrounding the application of XML to data models, one party will be referring to attributes in the context of XML and another to attributes in the context of data models, causing confusion.

XML Comments – The structure for inserting free text comments into XML. The same structure is used for SGML and HTML comments. <!-- *comment text here* -->

XML Component – A generic term used to refer to XML elements, attributes, and XML Schema *type* definitions.

(XML) Document - - [Paraphrased from the XML 1.0 specification] *"A data object is an XML document if it is well-formed, as defined in the XML 1.0, specification. A well-formed XML document may in addition be valid if it meets certain constraints"* as described by a schema. Synonymous with XML instance.

(XML) Elements – The fundamental unit of information in XML. Elements are encapsulated by tags, and may contain (among other things) attributes (declared inside the opening tag), other elements, or data.

(XML) Child Element – The hierarchical nature of XML allows elements to contain or be nested inside other elements, forming a conceptual data tree (see DOM). Often XML elements are referenced in terms of parent-child relationships. A child element is an element contained between the tags of a parent element. Child elements are also referred to as *descendants*, while parent elements may be referred to as *ancestors*.

XML Declaration – Every well-formed XML document must begin with a statement that at a minimum declares the version of XML that the document conforms to. Example: <?xml version="1.0">,

XML Document Tree – Refers to the logical model of an XML document conceptualized as a data tree, with a Root Node and branch nodes ending at data that can be thought of as the leaves. See DOM.

(XML) Grammar / Vocabulary – Related terms often used synonymously to indicate a set of element and attribute names and the structures described by a schema or set of related schemas that employ the elements and attributes. More precisely, the term vocabulary implies a commonly defined set of elements and attributes, while grammar refers to the composition of the vocabulary into meaningful business documents by one or more related schemas. An XML Namespace may be used to describe a vocabulary, while a schema may employ vocabulary from a single or multiple XML Namespaces.

(XML) Instance - Synonymous with XML Document. The term derives from object-oriented programming where objects are considered instances of classes. Programmers write code that defines application behavior in terms of classes of objects. In application execution, objects are *instantiated* (see object) from these class definitions. XML provides an object-like way to conceptualize textual data. Essentially, schemas are the equivalent of object classes, and XML documents are equivalent of object instances. Hence the term XML instance is widely used; however, XML document is the official term used by the W3C.

XML Namespace – An XML Namespace is a conceptual "space" to which element and attribute names may be assigned. An XML Namespace is declared within an XML instance by assigning a URI reference and an optional qualification prefix to an element. The element and all its children are considered to be "in" the XML Namespace unless specifically qualified with another Namespace's prefix. The URI reference does not have to an associated document physically at the URI. Within an XML Schema, the 'targetNamespace' attribute may be used to indicate that all elements declared within the schema are to be treated as "in" the target Namespace. The W3C Recommendation Namespaces in XML[lxi] provides the full specification for XML Namespaces. Note: DoD XML Namespaces may use XML Namespaces but the two terms are not synonymous.

(XML) Name Token – Per the XML 1.0 specification, a Name Token is "...any mixture of name characters..." where a "name" character obey the XML name convention. A [XML] *Name* "...is a token beginning with a letter or one of a few punctuation characters, and continuing with letters, digits, hyphens, underscores, colons, or full stops, together known as name characters. Names beginning with the string "xml", or any string which would match (('X'|'x') ('M'|'m') ('L'|'l')), are reserved for standardization in this or future versions of this specification." White space characters (hex #x20, #x9, #xD, #xA) are excluded from Name Tokens.

(XML) Parser - A software application (module) that either reads or receives a text encoded binary stream, decodes it, verifies the input conforms to "well-formedness" constraints of the XML 1.0 specification, (in the case of a Validating Parser) checks validity of the XML Instance against a schema if available, and exposes the content via an API to a calling application. A parser can be a standalone application, but it is most often a module called by a larger program (the calling application). A Parser may also be referred to as an XML Processor.

(XML) Processor - A synonym for an XML parser.

(XML) Registry – A web accessible application for registering information about XML components. Registration implies some degree of management and oversight. Registries collect and organize data about XML components ;they do not store the components themselves. XML component, schema and instance storage is the function of an XML Repository.

(XML) Repository – A web accessible storage mechanism for XML components. May or may not be associated with an XML Registry.

(XML) Root Node – The first node originating the XML Document Tree. The Root Node is not the same as the root element.

(XML) Root Element – Refers to the XML element in which all other elements must be nested. The root element (a physical XML construct) is a child of the logical root node of the document tree.

XML Schema Data Type – An XML component defined by the XML Schema language. Types do not show up in XML instances; they are used within the Schema to express relationships, and through type inheritance, add an object-like capability to XML Schemas. Types may be simple; that is, they allow definition of simple data-type constraints on element values, or they may be complex; that is, they define structures consisting of other elements. For example a type could be defined <***xsd:complexType name="AddressDetails"***>...</***xsd:complexType***>, then the definitions for XML elements, '***ShippingAddress***' and '***MailingAddress***' could reference the previously defined generic type.

(XML) Schema Annotation – The XML Schema language allows addition of annotations to schema components through an '***annotation***' element (<***xsd:annotation***>) which must contain either a '***documentation'*** element (<***xsd:documentation***>) or '***AppInfo****' element (<**xsd:appInfo**>). A '***source**' attribute may be added to either element to provide a URL reference to the source of the annotation. Annotations provide a more sophisticated way to provide documentation and application information that may be parsed and accessed by applications via an API.

(XML) Tags - XML (and its parent SGML) annotate metadata through the use of tags that indicate which text in a document are considered metadata and which is to be considered data. Tags are surrounded by markup characters. As an example, the data '3000N' can be marked up in XML, <latitude>3000N</latitude>. The tags are <latitude> (start tag) and </latitude> (end tag). Note: As discussed in the XML definition presented here, developers are free to defines tags. As an example, the data '3000N' could be alternatively marked up as, <lat>3000N</lat>, and still be well-formed. The document schema will specify which of all possible well-formed XML instances are valid for a particular application. An additional example is <***Latitude hemisphere="N"***>3000<***/Latitude***>; here the tag contains an XML attribute to specify the hemisphere. The choice as to the attribute name and possible values are also at the developer's discretion. Note that Parsers processing documents are sensitive to markup tag case; therefore, in the first example the tag <latitude> is not equivalent to the later example tag, <Latitude>.

XPath – XPath is a W3C recommendation whose primary purpose is to provide a compact, non-XML notation for identifying parts of an XML document. It operates on the abstract, logical structure of an XML document, rather than its surface syntax by modeling an XML document as a tree of nodes. The document tree can be navigated by applications implementing XPath. XPath is the result of an effort to provide a common syntax and semantics for functionality shared between XSL Transformations [XSLT] and XPointer.

XSL - The Extensible Style Sheet Language. [From the W3C XSL page[lxii]] "XSL is a language for expressing stylesheets. It consists of three parts: XSL Transformations[lxiii] (XSLT): a language for transforming XML documents, the XML Path Language[lxiv] (XPath), an expression language used by XSLT to access or refer to parts of an XML document (XPath is also used by the XML Linking[lxv] specification). The third part is XSL Formatting Objects: an XML vocabulary for specifying formatting semantics. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary. For a more detailed explanation of how XSL works, see the What Is XSL[lxvi] page." As of 16 October 2001, XSL[lxvii] is a W3C final recommendation.

XSL Processor - The software (module) executing XSL transformation and formatting instructions. At a minimum, consists of an XSLT conformant transformation component, and an optional XSL-FO processing component. A word of caution: XSL processor vendors often add "extensions" to the XSLT specification. While often extremely useful, stylesheets written using these extensions will not perform correctly in another XSLT compliant processor, eliminating their cross-platform compatibility.

XSL-FO - XSL Formatting Objects: an XML vocabulary for specifying formatting semantics. XSL-FO works in conjunction with XSLT to markup transformed XML with formatting object tags. Applications capable of processing these tags render the XML to another application's presentation environment. For example, Apache's Formatting Object Processor (FOP) can transform XML to Adobe PDF format. Another example is jfor, an open-source formatting object processor for transforming XML to Rich Text Format (RTF).

XSLT - XSL Transformations[lxviii] , a W3C recommendation [from the XSLT recommendation] *"…defines the syntax and semantics … for transforming XML documents into other XML documents*" [including well-formed HTML]." XSLT is the only W3C recommended XML syntax for transforming XML documents. Developers writing stylesheets should ensure they are strictly conformant to this specification to ensure reusability. Conformance testing through the use of several XSLT compliant XSL processors is recommended.

Appendix H – Implications of the XML Schema Language for XML Component Design

The following excerpt is taken from a draft document being produced as part of the OASIS Technical Committee developing the Universal Business Language. While the language here is oriented primarily towards the use of XML as a business-to-business messaging protocol, it provides a good description in business terms of the benefits of an XML Schema oriented approach to XML component design.

Implications of Schemas for Business Document Design

If we look at schema capabilities, certain considerations regarding data structure design strike us:

In existing XML schema languages, extensibility is largely limited to element content, and does not readily accommodate the modification of existing attributes on a particular XML element. Consequently, designers use elements rather than attributes to contain data that may be subject to extension in schemas.

Because data typing is much stronger when using XML schema processing, attention to the actual use of different kinds of data elements is critical in designing a common library. Where a DTD-based system would not produce errors over minor variations in the length of a #PCDATA field, for example, schema-validated XML applications will. The more control over our data our validation gives us, the more careful we need to be, or we will produce a standard data structure that will not be useful for some.

In many respects, as a result of schema extensibility, less is more. If we can identify those places within business document structure that are most liable to be extended, then we should model only the absolute common core. Because schema extension mechanisms are additive, it is better to recognize what is in fact common, rather than taking a (possibly wrong) guess at what might be useful.

Extensibility

The requirements of e-commerce are such that many basic document types are generally useful, but for specific tasks or for particular markets, minor structural variations are extremely useful. If a truly common XML structure is to be established for e-commerce, it will need to be easily modifiable, while minimizing the costs associated with implementation around these variations on standard data structures.

In EDI there has been a gradual increase in the number of different elements, to accommodate market-specific variations. Several efforts within the EDI community are focused on eliminating this problem, which points out the fact that variations are a requirement, and one that is not easy to meet. A related EDI phenomenon is the overloading of the meaning and use of existing elements, creating a tangible bar to interoperation without low-level coordination between trading partners. The end result is a high cost in implementation.

XML DTDs require that a data structure be described fully before implementation, in terms of its elements, attributes, and their structural relationships and content models. Without these fundamental structural rules in place, building an e-commerce application becomes difficult or impossible. For documents of a given document type to be interoperable across different e-commerce applications, they must conform to a single DTD, with only minimal variation in their structures. In practice, the high degree of cross-application coordination required to handle structural variation reduces the usefulness of this built-in document-specific capability of XML processing with DTDs.

Schema-based XML processing offers us a way to enhance the ability of applications to interoperate, because it accommodates the required variations in basic data structures without either overloading the meaning and use of existing data elements, or requiring wholesale addition of data elements specific to a particular industry or process. This is accomplished by allowing implementors to specify new element types that inherit the properties of existing elements. Schemas also allow you to specify exactly the structural and data content of the additions made to existing data structures. In this way, schemas allow us to limit variations and minimize the amount of additional implementation effort required in building an application.

This benefit derives from the nature of most variations required in e-commerce documents; many data structures are very similar to “standard” data structures, but have some significant semantic difference in a particular industry or process. Because schemas give us a mechanism for indicating the semantic “predecessors” of a particular variation, generic processing of standard types provides us with a basis for implementing just the refinements needed to handle the specific semantic variation. (An example of this would be the addition of a field to an address block to describe some industry-specific addressing information. The address structure could be taken from a common library. Only the single additional field would require new processing, even though the entire structure was given a different name to distinguish it from the “normal” address structure.)

In those cases where a variation in data structure is required only for some particular process, schemas again allow us to minimize implementation effort. It is possible to add a mechanism that allows a system to process a modified data element exactly as it would process its direct, standard parent, except for the specific interaction that requires the modified structure. By having most processes ignore the variation, except where it is specifically needed, schemas again help us reduce the effort required to build e-commerce applications and enhance the level of interoperability.

Note that schema syntax can express structural extensions and information about new data types. This ability can help users accommodate requirements placed on them by legacy processing systems with nonstandard specifications.

While the problems encountered in EDI applications cannot be avoided entirely, the use of XML schemas helps us identify variations in data structure and manage them better. Further, it gives us a solid syntax for modifying only those specific aspects of the data structure that require modification.

Modularity

Consideration was given to the usability of any standard set of e-commerce components. If we look at Simple-EDI, we have a case where the different types of elements have been formally classified:

Message Type—the type of the containing document/message

Segment—the type of the subsection (frequently nested)

Composite Data Elements—data elements that have both data members and some substructure

Data Elements—data elements without substructure

While Simple-EDI is organized according to this set of distinctions, XML, because it has a broader application, is not. In XML, an element at any level is potentially a substructure in some other element. In effect, a PurchaseOrder element is not significantly different than an AddressBlock element, even though their uses within a processing application may be very different. The generic processing capabilities of XML tools do not recognize any inherent difference.

In many ways, this capability of XML is advantageous. It allows us to process nested (“looping”) structures easily. It fails to provide any useful distinction about the functional roles played by any specific element in a particular XML application. If there is any formal distinction in XML, it is between mixed content elements. They can contain plain text as well as element substructures, and those elements whose only content is element substructures. Even here, the difference is not as clear as in EDI, because XML elements are capable of carrying attributes that always contain content.

However, when it comes to building a standard set of business documents that are easy to understand and use, the conceptual classification of data elements may be helpful. If such a classification is seen as useful, a four-level breakdown, based on the Simple-EDI model, would be the best approach. The WG recognized that this may or may not be helpful for a particular user population. As it is not a strong technical distinction in XML, this conceptualization is left up to those documenting a particular set of business documents for an e-commerce application. It is not seen as a necessary part of a standard business document set.

Description

XML Schemas can be broken into multiple schema documents, which can be assembled using includes and imports.

Benefits

- Smaller, modular schema documents encourage reuse.

- Smaller schema documents are easier to read and maintain.
- Schema documents can be used to organize schema components into logical units.

Risks

Breaking down schema documents too much (e.g. one schema document per type) can be confusing and inconvenient to users.

URL References

[i] Navy XML Quick Place, http://quickplace.hq.navy.mil/navyxml

[ii] Task Force Web, http://www.tfw.navy.mil/

[iii] RFC 2119, http://www.ietf.org/rfc/rfc2119.txt

[iv] XML Protocol Working Group, http://www.w3.org/2000/xp/Group/

[v] http://tis.eh.doe.gov/techstds/publaw.html

[vi] http://www.whitehouse.gov/omb/circulars/a119/a119.html

[vii] XML Schema Tutorial, http://www.xfront.com/xml-schema.html

[viii] XFront.com, http://www.xfront.com/

[ix] Schema Best Practices, http://www.xfront.com/BestPracticesHomepage.html

[x] eBTWG, http://www.ebtwg.org/

[xi] eBTWG UML2XML, http://www.ebtwg.org/projects/u2xdr.html

[xii] COE XML Registry, http://diides.ncr.disa.mil/xmlreg/user/index.cfm

[xiii] ebXML Specifications and Technical Reports, http://www.ebxml.org/specs/

[xiv] COE Data Emporium, http://diides.ncr.disa.mil/shade/refdatasets.cfm

[xv] MIL-STD-6040 (USMTF), http://www-usmtf.itsi.disa.mil/std_6040.html

[xvi] OASIS, http://www.oasis-open.org/

[xvii] UN/CEFACT, http://www.unece.org/cefact

[xviii] eBTWG, http://www.ebtwg.org/

[xix] ebXML Core Component Technical Reports, http://www.ebxml.org/specs/#technical_reports

[xx] ebXML Technical Architecture, http://www.ebxml.org/specs/ebTA.pdf

[xxi] ebXML Technical Report, Naming Convention for Core Components http://www.ebxml.org/specs/ebCCNAM.pdf

[xxii] W3C Technical Recommendations, http://www.w3.org/TR/

[xxiii] Business Process and Business Information Analysis Overview v1.0, http://www.ebxml.org/specs/bpOVER.pdf

[xxiv] ebXML Business Process Specification DTD, http://www.ebxml.org/specs/ebBPSS.dtd

[xxv] ebXML Business Process Specification XML Schema (CR), http://www.ebxml.org/specs/ebBPSS.xsd

[xxvi] COE XML Registry, http://diides.ncr.disa.mil/xmlreg/user/index.cfm

[xxvii] COE XML Namespace Managers, http://diides.ncr.disa.mil/xmlreg/user/namespace_list.cfm

[xxviii] COE XML Registration Information, http://diides.ncr.disa.mil/xmlreg/user/registry_info.cfm#submit

[xxix] The Object Management Group, http://www.omg.org/

[xxx] eBTWG Core Component Project, http://www.ebtwg.org/projects/core.html

[xxxi] DDDS, http://www-datadmn.itsi.disa.mil/ddds/ddds40.html

[xxxii] DoD 8320, http://www-datadmn.itsi.disa.mil/guidance.html

[xxxiii] COE XML Registry, http://diides.ncr.disa.mil/xmlreg/user/index.cfm

[xxxiv] COE XML Namespace Managers, http://diides.ncr.disa.mil/xmlreg/user/namespace_list.cfm

[xxxv] COE XML Registration Information, http://diides.ncr.disa.mil/xmlreg/user/registry_info.cfm#submit

[xxxvi] W3C DOM, http://www.w3.org/DOM/

[xxxvii] HTML, http://www.w3.org/MarkUp/

[xxxviii] OMG IDL, http://www.omg.org/gettingstarted/omg_idl.htm

[xxxix] ISO Store, http://www.iso.ch/iso/en/prods-services/ISOstore/store.htm

[xl] XML 1.0, http://www.w3.org/TR/2000/REC-xml-20001006

[xli] Namespaces in XML, http://www.w3.org/TR/1999/REC-xml-names-19990114/

[xlii] SAX, http://www.megginson.com/SAX/

[xliii] ISO 8879 (SGML), http://www.w3.org/TR/2000/#ISO8879

[xliv] XML Cover Pages - SOAP, http://xml.coverpages.org/soap.html

[xlv] SOAP 1.1, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[xlvi] SOAP 1.2, http://www.w3.org/TR/2001/WD-soap12-20010709/

[xlvii] W3C Process, http://www.w3.org/Consortium/Process-20010719/submission

[xlviii] UML, http://www.rational.com/uml/index.jsp

[xlix] Unified Modeling Methodology, http://www.gefeg.com/tmwg/tm090.pdf

[l] Rational Unified Process, http://www.rational.com/products/rup/index.jsp

[li] W3C Note, URI/URL/URN Clarification, http://www.w3.org/TR/2001/NOTE-uri-clarification-20010921/

[lii] W3C, http://www.w3.org/

[liii] W3C Members, http://www.w3.org/Consortium/#membership

[liv] W3C Consensus Processes, http://www.w3.org/Consortium/Process-20010719/submission

[lv] W3C Schema page, http://www.w3.org/XML/Schema

[lvi] W3C Activity Statement, http://www.w3.org/XML/Activity.html

[lvii] XML Schemas: Part 0, http://www.w3.org/TR/xmlschema-0/

[lviii] XML Schemas: Part 1, http://www.w3.org/TR/xmlschema-1/

[lix] XML Schemas: Part 2, http://www.w3.org/TR/xmlschema-2/

[lx] XHTML, http://www.w3.org/MarkUp/#xhtml1

[lxi] Namespaces in XML, http://www.w3.org/TR/1999/REC-xml-names-19990114/

[lxii] W3C XSL Page, http://www.w3.org/Style/XSL/

[lxiii] XSL Transformations, http://www.w3.org/TR/xslt

[lxiv] XPath, http://www.w3.org/TR/xpath

[lxv] XLink, http://www.w3.org/TR/xlink/

[lxvi] What is XSL, http://www.w3.org/Style/XSL/WhatIsXSL.html

[lxvii] XSL Final Recommendation, http://www.w3.org/TR/2001/REC-xsl-20011015/

[lxviii] XSLT, http://www.w3.org/TR/xslt