Department of the Navy XML Naming and Design Rules

Office of the DON
Chief Information Officer

Final Version 2.0
January 2005

Department of the Navy XML Naming and Design Rules

January 2005

DEPARTMENT OF THE NAVY
CHIEF INFORMATION OFFICER
1000 NAVY PENTAGON
WASHINGTON, DC 20350-1000

18 January 2005

MEMORANDUM FOR DISTRIBUTION

Subj: EXTENSIBLE MARKUP LANGUAGE (XML) NAMING AND DESIGN RULES OFFICIAL RELEASE

To comply with joint requirements as embodied in the DoD Net-Centric Data Strategy and to achieve the FORCEnet requirement for a common structure and language for information handling, the Department of the Navy (DON) is issuing Naming and Design Rules (NDR) that facilitate the discovery and use of common data across the naval enterprise. XML, an open standards based technology, is a key enabler of the Department's net-centric data strategy.

The NDR provides additional rigor necessary to efficiently and effectively operate in a net-centric data-sharing environment. These rules move the DON forward to ensure that all XML is based on a consistent set of schema through the application of open standards that align with the Federal Enterprise Architecture Data Reference Model and Global Information Grid. The result will be an environment that is sustainable, responsive, and agile.

The NDR is the product of expertise and energies contributed by representatives from 13 key Navy, Marine Corps, and Secretary of the Navy organizations who participated in the DON XML Working Group. To ensure these rules are applicable and current, the DON Chief Information Officer (DON CIO) has established the XML Business Standards Council and is proceeding to charter the Net-Centric Technical Standards Council to serve as liaison to organizations developing national and international standards for XML and Web Services technologies.

All commands in the Navy and Marine Corps that are developing systems that use XML, in accordance with our DON XML Policy, should apply these standards in order to maximize interoperability and enable a net-centric environment for enhancing supportability of operations across the Department.

The DON XML NDR can be downloaded from the web at www.doncio.navy.mil. If you have questions concerning DON XML policy and guidance, or would like to participate in our DON XML initiatives, you can contact us from the DON CIO webpage.

D. M. Wennergren

Contents

Table

Figures

Section 1
Introduction

The Department of Navy (DON) vision for Extensible Markup Language (XML) is "to fully exploit XML as an enabling technology to achieve interoperability in support of maritime superiority."[1] In accordance with this vision, the DON developed this document, *Department of Navy XML Naming and Design Rules,* Version 2.0 (referred to in this document as the DON NDR), to provide the DON XML developer with a clear and comprehensive set of XML development rules and guidance that will standardize XML across the DON and promote global interoperability. In keeping with the tenets of federal statute[2] and policy,[3] this document utilizes existing voluntary consensus standards (VCS) to the maximum extent practical.

A set of naming and design rules (NDR) is necessary because many different development options are available for creating XML schema. For example, as a direct result of the inherent flexibility within the XML Schema standard developed by the World Wide Web Consortium (W3C), a developer can create an XML schema in one DON organization that is incompatible with a schema from another DON organization.[4] Thus, these two organizations would not be able to successfully exchange data using XML. Both schemas, however, are perfectly valid with respect to the W3C Schema standard.[5] By publishing a comprehensive set of NDR, the DON aims to eliminate the potential for XML incompatibility.

The DON NDR will immediately provide two tangible benefits. First, the DON NDR will standardize the development of XML schema across the DON. Second, and more important, the DON NDR will provide an XML information analysis framework that leverages the syntax neutral object model specified by the Core Component Technical Specification (CCTS).[6] The CCTS is published by UN/CEFACT in conjunction with OASIS as Part 8 of the ebXML framework.[7] The CCTS is experiencing widespread adoption in the national and international public and private sectors as the preferred approach for data modeling.

[1] Department of Navy Vision for Extensible Markup Language (XML), 15 March 2002.

[2] The National Technology Transfer and Advancement Act of 1995, PL 104-113, 7 March 1996.

[3] Federal Participation in the Development and Use of Voluntary Consensus Standards and in Conformity Assessment Activities, OMB Circular A-119, 10 February 1998.

[4] W3C XML Schema Specification Parts 1 and 2, 2 May 2001.

[5] See www.w3.org.

[6] UN/CEFACT, Core Components Technical Specification: Part 8 of the ebXML Framework, 15 November 2003, Version 2.01.

[7] See http://www.ebxml.org/.

The CCTS is based on ISO 11179: Information Technology—Metadata Registries: Naming and Identification Principles for Data Elements Part 5. ISO 11179 is also specified by the Federal Enterprise Architecture (FEA) Data Reference Model (DRM).

The DON NDR specifies the CCTS as the basis for DON XML information analysis. Developers should read the CCTS for a more thorough understanding of the methodology and CCTS terminology (ABIE, BBIE, BIE, etc.) used throughout this document. The definitions of most of these terms are in the glossary of this document, but they are better understood through reading the CCTS.

Core components provide a syntax neutral representation of data elements that will enable global interoperability across the DON by standardizing the way developers name data elements.[8] The CCTS provides detailed rules for documenting data elements by creating units that are semantically unambiguous and interoperable. CCTS core components are interoperable units of data and will ultimately become the XML data elements that are exchanged among DON organizations.

The other key international XML standard relevant to the development of the DON NDR is the Universal Business Language (UBL) specification, developed by the Organization for the Advancement of Structured Information Systems (OASIS). OASIS developed the UBL NDR based on the CCTS; the UBL NDR describes how to implement CCTS in XML schema.[9] The OASIS UBL rules provide a foundation for the DON NDR. In addition, by using the OASIS UBL rules, the DON can leverage the significant work that the OASIS Technical Committee (TC) has contributed to develop an XML standard for global interoperability.

Finally, the DON NDR addresses many key aspects of XML schema architecture that are critical to effecting successful data management and harmonization within the DON. We further detail these topics, including the XML data management approach, XML namespace and modularity, and versioning, in this document.

1.1 Purpose

The purpose of this document is to summarize the DON NDR approach for every main XML rule category, list the associated XML rules, and provide adequate context and examples to understand the rules. The Business Standards Council (BSC) approved this document after significant research, analysis, and discussion.

1.2 Authority

The DON NDR has been developed under the authority of the DON XML governance structure and in accordance with the DON XML vision and policy.[10,11] All XML rules

[8] "Syntax neutral" refers to the separation of metadata from specific markup or message syntaxes, like XML or Electronic Data Exchange (EDI), used to express data and metadata in messages or content.

[9] See http://oasis-open.org/committees/ubl/.

[10] *Department of Navy Vision for Extensible Markup Language (XML)*, 15 March 2002.

[11] *DON Policy on the Use of the Extensible Markup Language (XML)*, December 2002.

contained herein have been approved by the DON XML governance structure and are effective immediately. The DON XML NDR Version 2.0 supercedes *all* previously released naming and design rules, including *DON XML Developers Guide Version 1.0* and *DON XML Developers Guide Version 1.1*.

1.3 Audience

The primary audience for this document is the DON developer who is currently developing IT solutions that employ XML for data exchange or publication. W3C's XML Schema standard provides a broad range of flexibility. This flexibility offers the designer many choices—global versus local elements, use of built-in simple types or creation of user-defined simple and complex types, and derivation of types and elements by extension or restriction, to name a few. The DON NDR provides authoritative guidance for all uses of XML in the DON. The DON NDR will also provide authoritative guidance for the DON Functional Namespace Coordinators (FNCs) and the Technical Assistance Team (TAT).

1.4 Scope

The scope of this document is to provide a comprehensive set of design rules and naming conventions for the creation of all XML schema and instances across the DON.[12] The DON NDR guidance is applicable to all DON organizations and commands that use XML, including all commercial and government off-the-shelf product implementations.

1.5 NDR Document Versioning

This document is versioned as Version 2.0. The initial release of the DON naming and design rules was *DON XML Developers Guide Version 1.0* (29 October 2001). The subsequent release, *DON XML Developers Guide Version 1.1*, followed the same naming convention. The name change was necessary to release a subset of the DON XML developers guide, before completing the comprehensive developers guide. Thus, the *DON NDR* is versioned as Version 2.0 and will be incorporated into the *DON XML Developers Guide Version 2.0* (or subsequent version) thereafter.

1.6 Guiding Principles

The DON developed the DON NDR based on a set of guiding principles derived from those developed by the OASIS UBL Technical Committee. The DON XML guiding principles encompass three areas:

- General UBL guiding principles
- Extensibility
- Code generation.

[12] Exceptions are allowed for instances where DoD policy may supercede department level policy.

The DON XML guiding principles are as follows:

1. *Internet Use.* DON XML shall be straightforwardly usable over the Internet.

2. *Broad Applicability.* The possible uses of XML within the DON are numerous and impossible to completely predict. Therefore, DON NDR shall be sufficiently broad to incorporate a wide variety of XML uses. Rules shall allow for both data interchange, as well as the publication of content, both static and dynamic. The rules shall not, however, address the creation of XML instances that are used solely within a software system to carry technical (rather than business or mission execution) information pertinent only to the internal functioning of the system.

3. *Tool Use and Support.* The design of DON XML schemas will not make any assumptions about sophisticated tools for creation, management, storage, or presentation being available. The lowest common denominator for tools is incredibly low (for example, Notepad), and the variety of tools used is staggering. We do not see this situation changing in the near term.

4. *Legibility.* DON XML documents should be easy to understand and reasonably clear.

5. *Simplicity.* The design of UBL must be as simple as possible.

6. *80/20 Rule.* The DON NDR should provide the 20 percent of features that accommodate 80 percent of the needs.

7. *Component Reuse.* The design of DON XML reusable XML components should contain as many common features as possible. The range of possible uses of XML within the department is extremely broad; the purpose of the NDR is to provide a basis for XML schemas and instances produced by different organizations to achieve similarity and therefore share common semantics.

8. *Standardization.* The number of ways to express the same information in a DON XML schemas document is to be kept as close to one as possible.

9. *Domain Expertise.* The DON XML Naming and Design Rules document development will leverage expertise in a variety of domains through interaction with appropriate development efforts.

10. *Customization and Maintenance.* The design of DON XML schemas must facilitate customization and maintenance.

11. *Context Sensitivity.* The design of DON XML schemas must ensure that context-sensitive document types are not precluded.

12. *Prescriptiveness.* The DON NDR design will balance prescriptiveness in any one usage scenario with prescriptiveness across the breadth of usage scenarios supported. Having precise, tight content models and datatypes is a good thing

(and for this reason, we might want to advocate the creation of more document type "flavors" rather than less; see below). However, in an interchange format, it is often difficult to get the prescriptiveness that would be desired in any one usage scenario.

13. *Data versus Document Orientation.* The DON NDR must provide the capability to express both content and elements of data in a single document. This must be accomplished through reuse of existing BSC-approved document-structure or presentation-oriented XML vocabularies within instances of XML containing elements structured according to the NDR. Document oriented data (tags) (paragraph, chapter, list, etc.) will not be used to capture data elements that may be used for other data exchange or data mining.

14. *XML Technology.* The DON NDR will avail itself of standard XML processing technology wherever possible (XML itself, XML schema, XSLT, XPath, Xlink, and so on). However, the DON will be cautious about basing decisions on "standards" (foundational or vocabulary) that are works in progress.

15. *Associations and References.* Because the scope of possible uses of XML within the DON is very broad and impossible to predict, the DON NDR must be flexible enough to accommodate a wide range of mechanisms that can be used to express relationships between XML elements in the same document or between elements in different documents. Rules must be developed to allow instances of XML to be normalized (not contain redundant markup) and to allow elements to reference markup in other XML documents.

16. *DON Enterprise Namespace.* The DON XML policy has stated that there will be one DON XML enterprise namespace. The DON NDR shall facilitate the development of XML components by organizations such that they can be easily migrated to this namespace upon harmonization.

17. *Relationship to Other Non-DON Namespaces.* The DON NDR will allow for reuse of markup from other controlled XML vocabularies designated by the DON BSC as allowable. In addition to reuse of Root-Level Schema modules from approved vocabularies, the NDR must provide for a mechanism to reuse elements and attributes from these controlled vocabularies directly in instances structured according to the DON rules.

18. *Adherence to DON Policy.* These NDR must be consistent with the requirements and spirit of both the DON XML policy and SECNAVINST 5000.36. Accordingly, the rules must recognize that they are specifying both metadata (the purview of FDMs) and XML binding of that metadata (the purview of FNCs).

19. *Adherence to DoD Policy.* The DON rules must be consistent with existing, and future, overarching DoD guidance and policy. Specifically, the DON NDR must comply with higher-level department policy and guidance that potentially may impact DON XML data management and design rules that are applicable.

1.7 Document Organization

This document is organized into sections, each covering one of the following nine topics:

- Information analysis rules
- Data management rules
- Namespace and schema modularity rules
- Versioning rules
- General XML rules
- Naming, definition, and declaration rules
- Code list rules
- XML instance rules.
- Security rules.

The appendixes are listed below:

- Appendix A—Rules Tables
- Appendix B—Bibliography
- Appendix C—Glossary/Acronyms
- Appendix D—Schema Examples
- Appendix E—Approved XML Specifications.

Each of these sections has standalone, sequentially numbered subsections. As other issues are uncovered in the future, particularly those relating to interoperability, the DON XML BSC or its delegates will investigate them and add the new areas as separate sections or subsections when applicable.

1.8 Terminology and Notation

In this document, the key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL are to be interpreted as described in Internet Engineering Task Force (IETF) Request for Comments (RFC) 2119. Lowercase forms of these words are used in the regular English sense.

The following style formats were used to represent various content types including examples, notes and rules. The examples contained in this document may not validate without supporting code. Examples are for illustration only.

Example: *Title of the Example*

A representation of a rule and they are informative.

[NOTE] Explanatory information. Notes are informative.

Rule

[Rule Prefix & Number]–see below*

*[RRRn]—Identification of a rule that requires conformance to ensure that an XML schema is UBL conformant. The value RRR is a prefix to categorize the type of rule, where the value of RRR is as defined in Table 1-1 and n (1.n) indicates the sequential number of the rule within its category. To ensure continuity across versions of the specification, rule numbers that are deleted in future versions will not be reissued, and any new rules will be assigned the next higher number, regardless of location in the text. Future versions will contain an appendix that lists deleted rules and the reason for their deletion. Only rules are normative; all other text is explanatory.

Table 1-1. Rule Prefix Token Value

Abbreviation	Rule
ATD	Attribute Declaration
CDL	Code List
CTD	ComplexType Definition
CTN	ComplexType Naming
DOC	Documentation
ELD	Element Declaration
ELN	Element Naming
GNR	General Naming
GXS	General XSD
IND	Instance Document
INF	Information Analysis
MDC	Modeling Constraints
NMS	Namespace
RED	Root Element Declaration
SEC	Security
SSM	Schema Structure Modularity
STA	Standards Adherence
VER	Versioning

The terms “W3C XML Schema” and “XSD” are used throughout this document. They are considered synonymous; both refer to XML schemas that conform to Parts 1 and 2 of the W3C XML Schema Definition Language Recommendations. See the glossary (Appendix C) for additional term definitions.

Section 2
Information Analysis Rules

Information analysis is a critical and necessary precursor to successfully employing the CCTS syntax neutral model across the DON. Throughout the NDR, the UN/CEFACT Modeling Methodology (UMM) was used to model the business processes upon which the corresponding XML examples are based.[1] The DON NDR recommends modeling business processes using a standard approach (e.g., UMM or an equivalent business process modeling methodology) to develop logical data models. All XML data models, however, must incorporate Business Information Entities in accordance with the CCTS syntax neutral model.

In general, after DON XML information analysis and data modeling are completed, the developer will use the resulting information model to create XML schema in accordance with the DON NDR. DON XML data modeling must be flexible enough to allow existing frameworks—e.g., Department of Defense Architecture Framework (DODAF) data frameworks—to coexist with DON XML framework models. The DON NDR, however, strongly recommends using the UMM or an equivalent information analysis framework when creating XML data models.

This section contains the DON rules and guidance for XML information analysis, data modeling constraints, and schema creation that closely adhere to the CCTS and UBL framework standards. This section does *not* include a tutorial for how to use Unified Modeling Language (UML).

2.1 Purpose

The purpose of this section is to summarize the DON XML information analysis approach and the development of the syntax neutral model. The information analysis process produces a set of data elements documented with appropriate metadata, such as CCTS component types. CCTS component types must be Aggregate Business Information Entities (ABIEs), Association Business Information Entities (ASBIEs), Basic Business Information Entities (BBIEs), or DataTypes (qualified or unqualified).

2.2 Aggregate (ABIE) and Association (ASBIE) BIEs

An ABIE is a collection of related pieces of business information that together convey a distinct business meaning in a specific business context. Expressed in modeling terms, it is the object class, in a specific business context.

[1] The term "business processes" refers to the work processes at all DON organizations and activities including the business, war fighter and intelligence community domains.

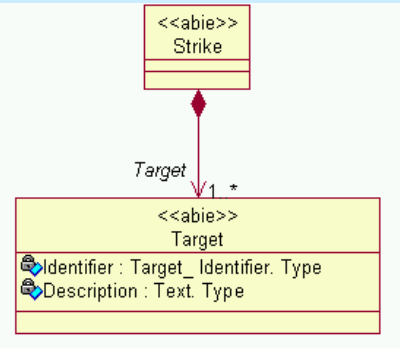
An ASBIE is a business information entity that represents a complex business characteristic of a specific object class in a specific business context and is the association to another ABIE. Thus, an ASBIE is a property of one ABIE that associates it to another. An ASBIE must include the following three-part dictionary entry name:

- Object class = object class term of the associating model class.
- Property term = role name of the association.
- Representation term = object class term of the model class being associated to.

Examples of several different types of associations are provided below.

In this example, the relationship between "strike" and "target" is expressed as "A strike consists of one or more targets."

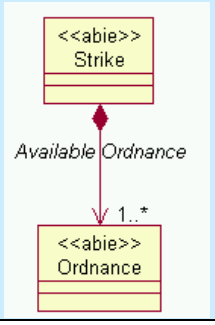
Example: *Association to an ABIE→ASBIE—Strike. Target*



In this example, the "Strike. Details" ABIE has an ASBIE "Strike. Target." The property and representation term of the dictionary entry name are both "target." When terms are duplicative, the dictionary entry name "Strike. Target. Target" can be truncated to "Strike. Target."

Example: *Association to an ABIE using a **role** name*

ASBIE—Strike. **Available**. Ordnance

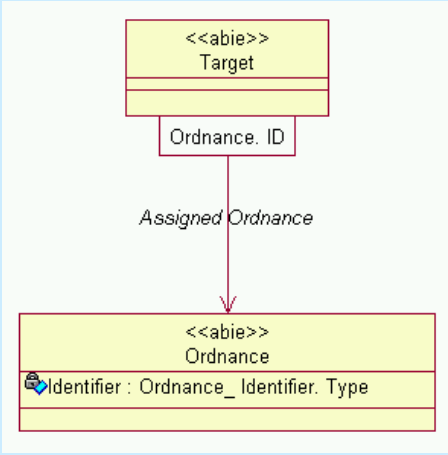


In this example, the "Strike. Details" ABIE has an ASBIE "Strike. Available. Ordnance." The role name of the association "Available" adds meaning to the ordnance class. Compare this to the first example in which the association name and the class being associated to are the same.

2.2.1 Reference Associations

Reference associations follow the same rules as the general associations. Reference associations, however, provide the ability to normalize the association relationship between two ABIEs.

Example: *Reference Association ASBIE–Target. Assigned. Ordnance*



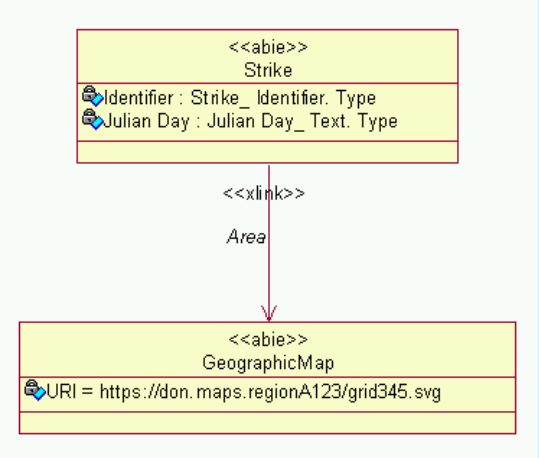
In this example, target "Target. Details" and ordnance "Ordnance. Details" are associated by reference or unique identifier. The reference is expressed in UML as an association qualifier "Ordnance. ID." The use of an association reference allows a developer to normalize instances of ordnance using XML schema.

The DON NDR contains rules for expressing reference associations using the XML Schema identity constraints xsd:unique and xsd:keyref. The association shown here is one to one (1:1). Reference associations may also be one-to-many (1:n).

2.2.2 X-Link Associations

The DON NDR allows XLink associations between model objects that are linked to other information resources.

Example: *XLink Association ASBIE–'Strike. Area. Geographic_ Map'*

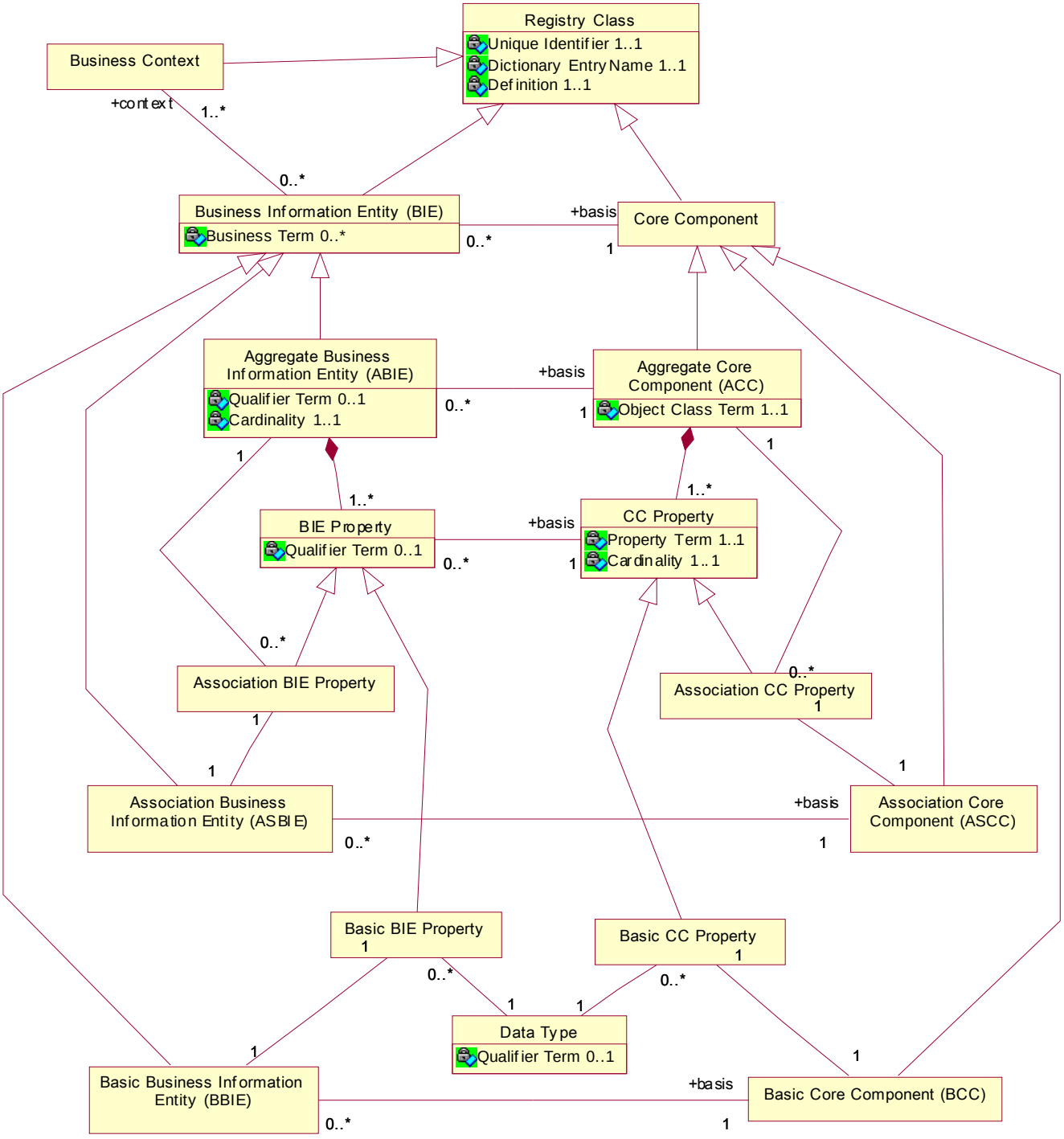


In this example, the association is between the strike class and a geographic map of the area in Scalable Vector Graphic (SVG) format. The resource is located at https://don.maps.regionA123/grid345.svg. The DON NDR for binding this association for XML will employ the Extensible Linking Language (Xlink) and will be further illustrated in the Element Declaration (ELD) and Element Naming (ELN) rules.

2.3 Basic Business Information Entities (BBIE)

A BBIE is a business information entity that represents a singular business characteristic of a specific object class. BBIEs are singular or simple properties that have one business value. BBIEs also have "BBIE Properties," as shown in Figure 2-1.

Figure 2-1. CCTS Object Model



Source: UN/CEFACT, Core Components Technical Specification: Part 8 of the ebXML Framework, 15 November 2003, Version 2.01.

This UML class diagram illustrates that ABIEs contain BIE properties, which may be a BBIE or ASBIE properties. It further illustrates that the BIE properties are associated but

separate from the actual BBIE. The distinction between a BBIE and a BBIE property is subtle but important for reuse. Consider the following example:

Example: *Sample BBIEs*

'Ordnance. Description. Text'
'Target. Description. Text'

Both of the BBIEs, shown in the previous example, share a common property and representation term, "Description. Text." The DON NDR refers to "Description. Text" as a BBIE property of either "Ordnance" or "Strike" and does not become a BBIE until it is referenced within an object class.

2.4 Qualified and Unqualified Datatypes

Unqualified Datatypes (UDTs) are used to derive elements. UDTs are types that are based from the CoreComponentType schema module (CCT.xsd). The CCT module is a normalized Schema that represents all required CCTS datatypes. Qualified Datatypes (QDTs) are derived from the UDTs.[2] Facets can be used to restrict UDTs to match the appropriate use.

Example: *BBIE Data Types and the UDTs from which they are derived*

Person. Occupation. Text (UDT = "Text. Type")
Person. First. Name (UDT = "Text. Type")

The BBIE "Person. Occupation. Text" uses the representation term "Text" in the representation term of its dictionary entry name. "Text" is a primary representation term based on the UDT "Text. Type." In the second example, "Person. First. Name," "Name" is a secondary representation term also based on the CCT "Text. Type."

To ensure that all data elements share a common semantic underpinning, the set of fundamental types must be limited. The CCTS defines these fundamental types as CCTs. The DON Enterprise UDT Schema module contains all of the fundamental CCTS types that are contained in the CCT Schema module.

Modeling Constraints Rule	
[MDC1]	DON XML components MUST be based on approved ccts:UnqualifiedDataTypes that are based on ccts:CoreComponentTypes.

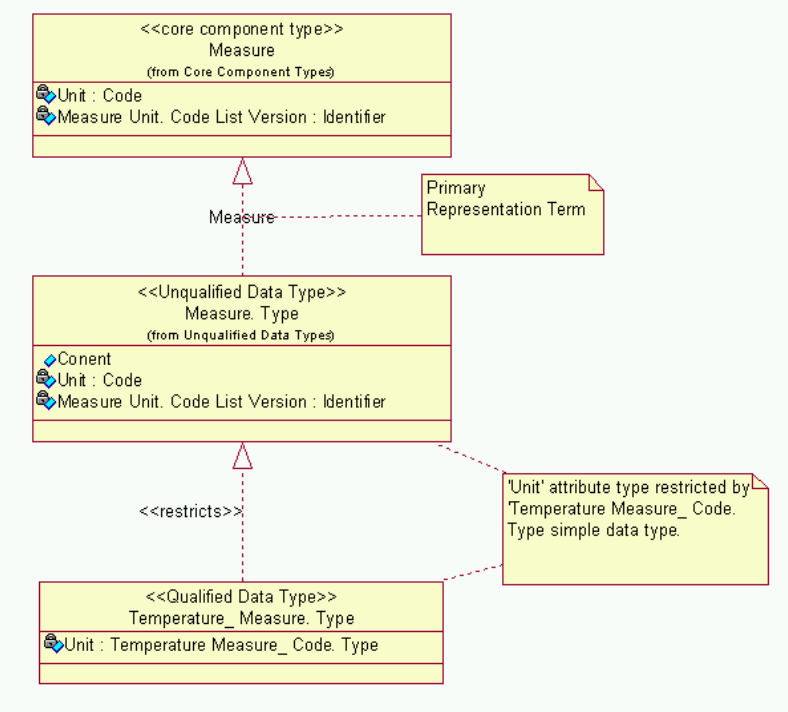
The DON NDR has adopted the convention of referring to the set of allowable CCTS representation terms that are based on underlying UDTs.

To promote reuse, data types may be extended or restricted to create new qualified data types. The CCTS allows the use of qualifiers as "a word or group of words that help define and differentiate an item from its associated items." Qualifiers are added to unqualified data types to differentiate them from the CCTs upon which they are based.

[2] Appendix E includes more detailed Schema examples of how QDTs are derived from UDTs.

Thus, a QDT is a ccts:DataType with a qualifier term appended. QDTs may be based on unqualified data types or other qualified data types.

Example: *Data Type Class Diagram*



This UML Class diagram expresses the relationship between the **Core Component Type** "Measure", the **Unqualified Data Type** "Measure. Type", and a **Qualified Data Type** 'Temperature_ Measure. Type'.

Note: The CCT Schema module is a normalized Schema for reference only. The CCT will not be imported into the DON Enterprise. The DON UDT Schema module includes all CCTS fundamental types.

Information Analysis Rule	
[INF1]	A qualified ccts:DataType must be created in the Syntax Neutral Model for each BBIE that uses domain restriction.

Developers must create a qualified data type for each BBIE that has a domain restriction over and above those imposed by the base UDT.

2.5 Developing the Syntax Neutral Model

Typically, the syntax neutral model is nothing more than a table, or spreadsheet, in which each row is a data element and each column documents metadata required for that element (e.g., dictionary entry name). CCTS specifies minimum required metadata for each type of model element.

This simplified two-dimensional model may not be adequate for more complex applications. The UMM provides a set of model diagrams that allow graphical representation of more complex relationships among model objects.

To promote interoperability, DON information models must define model elements consistently.

Modeling Constraints Rule	
[MDC2]	DON information models MUST define classes based on Aggregate Business Information Entities (ABIEs) and ccts:DataTypes.

The term "class" refers to the idea of an object class as a discrete unit of information represented by a graphical element. A class is equivalent to an entity in entity-relationship modeling. Classes are *not* defined for BBIEs or ASBIEs.

Example: 'Strike. Details' *Aggregate Business Information Entity (ABIE) and the following components:*

ABIE—*Strike. Details*

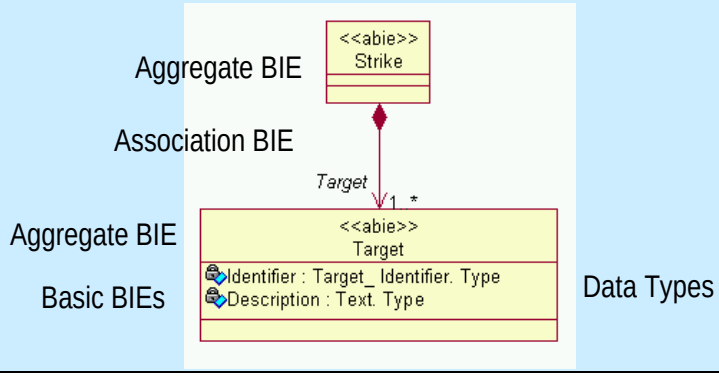
ASBIE—*Strike. Target*

BBIEs—*Target. Identifier, Target. Description. Text*

QDT—*Target_ Identifier. Type*

UDT—*Text. Type*

The following UML class diagram contains the 'Strike. Details' ABIE (Class), 'Strike. Target' ASBIE 'Target. Identifier', 'Target. Description.Text' BBIEs, 'Target_ Identifier. Type' QDT and 'Text. Type' UDT:



A developer can extend or restrict a data type to create a new type with different domain constraints. When developers modify a data type, they must maintain the original business context of the original data type.

Modeling Constraints Rule	
[MDC3]	If a DON ccts:DataType is extended or restricted, it MUST retain the original business context.

Example: *Maintaining Business Context*

'Target. Identifier' vs. 'Ordnance. Identifier'

An xsd:complexType created to represent "Target. Identifier" must not restrict or extend an xsd:complexType representing "Ordnance. Identifier." The business context of these identifiers is different.

2.6 Mixed Content

In the course of analysis and modeling, the contents of a particular model element may be defined by another BSC-approved XML business standard. When this occurs, DON analysts need not incorporate into their model elements from this other standard. The General XML Schema Rules (GXS) provide a mechanism for specifying the objects defined by another BSC-approved XML framework using the following expression:

Example: *Allowable Mixed Content Expression*

<xsd:any namspace="someURI">

Modeling Constraints Rule	
[MDC4]	Mixed content MAY only be used when an XML schema component is defined by a namespace from a BSC-approved business standard (e.g., XHTML).

Mixed content occurs when an XML data model has both data and other document values. In XML, an element can contain a combination of both text and markup. For example, the DON NDR allows for mixing text and markup when using the Extensible Hypertext Markup Language (XHTML).

Section 3
Data Management Rules

The W3C XSD was designed to be flexible enough to satisfy a diverse set of user requirements in a wide range of organizational and trading partner environments.[1] This flexibility offers the designer many choices—global versus local elements, use of built-in simple types or creation of user-defined simple and complex types, and derivation of types and elements by extension or restriction, to name but a few. However, this flexibility has certain drawbacks, primarily when attempting to use XML in a manner that enhances rather than detracts from interoperability, or more succinctly, when attempting to ensure consistency in XML implementations within a particular organization or enterprise.

The DON views XML as a key enabler in achieving enterprise interoperability and promulgating authoritative source data. Achieving these goals using XML requires establishing rules for consistent XML schema creation by all DON stakeholders. These rules must provide concise and clear guidance on the design and use of XML. More important, these rules must clearly articulate the which, why, and how DON XSDs will be designed. Chief among the design decisions the DON must address is how to declare XML elements.

3.1 Purpose

The purpose of this section is to summarize the DON XML data management approach.

3.2 Global Element Data Management

After significant research, analysis, and discussion, the BSC approved a global element approach to data management. A global element approach focuses on both individual element and object level information, creating global data elements for *all* objects (XML Complex Types) *and* object properties. This approach allows the following local XML elements to represent data elements that have been approved by the cognizant FNC and BSC:

- Identifier
- Code
- Measures.

[1] The terms "XSD," "XSD Schema," and "Schema" are interchangeable; all refer to a Schema developed in full conformance with the W3C Recommendation *XML Schema Part 1: Structures*, 2 May 2001, and W3C Recommendation *XML Schema Part 2: Datatypes*, 2 May 2001.

Element Declaration Rule (Reference ELD1 in General Element Declarations)

[**Ref**-ELD1] All element declarations MUST be global with the exception of `Identifiers, Measures, and Codes`, which MAY be declared as local elements if, and only if, approved by the FNC and BSC.

The FNC and BSC will approve identifiers, codes, and measures as local elements based on more detailed selection criteria (contained in later NDR sections).

Global elements are declared as direct children of a root schema element. Global element names, because they are globally reusable, express universally unambiguous semantics in their names.[2] By their nature, global elements can be reused consistently across the DON enterprise XML domain, wherein every occurrence will have exactly the same meaning and map to exactly the same authoritative source data. Therefore, the DON approach uses global elements in conjunction with global complex types.

Creating globally unique elements requires the application of a rigorous methodology specifically designed for this task. Fortunately, such a methodology already exists in the form of an international standard issued by the International Organization for Standardization (ISO): ISO 11179, *Information Technology—Specification and Standardization of Data Elements: Parts One through* Six.[3,4] In general terms, ISO 11179 methodology calls for an element's construct and name to contain three parts—object class and qualifiers, property term and qualifiers, and representation term. This is, from earlier discussion, the dictionary entry name.

The following example illustrates the principle. Each element carries a very precise name (all three parts of the ISO 11179 convention).

Example: *The Three Parts of the ISO 11179 Convention*

```
<Target>
        <TargetTargetID>…</TargetTargetID>
        <TargetName>…</TargetName>
    <TargetDescription>…</TargetDescription>
</Target>
```

In this example, the globally named leaf node BBIE (representing TargetID, Name and Description properties of a Target object class) repeat the object class name in each element.[5] Global elements are highly desirable from an enterprise interoperability and

[2] For example, they can be reused directly by any other element or complex type.

[3] UN/CEFACT, *ebXML Core Components Technical Specification*, Version 2.01, 15 November 2003.

[4] The DON is defining XML in the context of the UN/CEFACT ebXML CCTS. Under this concept, global elements are equivalent to Business Information Entities (BIEs). The CCTS defines BIEs as "a piece of business data or a group of pieces of business data with a unique business semantic definition."

[5] The Draft XML Development Guide Version 2.0, Volume 1, Glossary and Guideline, G-XG-1, defines or provides references to authoritative definitions of terms and concepts regarding basic XML structural components such as Leaf Nodes.

semantic clarity perspective because they do not allow two data elements with the same name, in the same namespace, to coexist.

The DON approach contains all of the benefits of global elements, while simultaneously addressing the redundancy issue. This approach, developed by the OASIS UBL TC of XML naming and design rules, uses a combination of fully qualified dictionary entry names and *truncated* global element names that take advantage of XML's unique object inheritance feature.

Under the UBL approach, the global element declarations include only the property term and qualifiers, and representation term components of ISO 11179 names for leaf node (CCTS based BBIE XML elements).[6] Therefore, as long as the element's definition is sufficiently broad to be applicable to a wide range of possible object classes, these elements may be declared globally. XML schema complex types representing ABIEs (carrying object class semantics in their name) include "by reference" the leaf node (BBIE) global XML elements (carrying property and representation term semantics). The globally defined BBIE elements in effect "inherit" the object class part of their semantics from the complex type that reuses them. This approach requires the leaf node (BBIE) elements to be defined very carefully so they are globally unique even without an object class. Certain BBIE XML elements—such as identifiers, codes, measures, and associations between object classes—may become overly complex with this approach.

In the following example, the global XML elements 'TargetID' and 'Description' represent BBIEs with ISO 11179 property terms and representation term pairs of Target. Identifier and Target. Description. There dictionary entry names "inherit" their object classes (Target) as a result of their inclusion by reference as children of the 'Target' ABIE, however, the object class (Target) is not included in the tag name.

[6] Where necessary, object class qualifiers may also be appended to the element name.

Example: Global Element Schema with Complex Type (Example shown does not include all required documentation for the purposes of this illustration.)

```
<!—Global Element—>
<xsd:element name="Target" type="TargetType"/>

<!—Global ComplexType—>
<xsd:complexType name="TargetType">
      <xsd:sequence>
            <xsd:element name="TargetID" type="TargetIDType"/>
            <xsd:element ref="Description"/>
            <xsd:annotation>
                <xsd:documentation>
                <cdp:DictionaryEntryName>Target. Description.
Text</cdp:DictionaryEntryName>
                <cdp:ObjectClass>Target</cdp:ObjectClass>
                <cdp:PropertyTerm>Description</cdp:PropertyTerm>
                <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:element ref="AssignedOrdnanceREF"/>
            <xsd:element ref="CountryCode"/>
            <xsd:element ref="SurfaceTemperatureMeasure" minOccurs="0"/>
      </xsd:sequence>
</xsd:complexType>
```

Eliminating potential barriers to DON XML system interoperability is a key driver for choosing the global element approach. The advantages of this approach become more evident when considering interdepartmental XML interoperability and data exchange. Because the recommended approach relies on global elements to carry unique semantics, and only one global element can exist for each globally defined BBIE, there cannot be a duplicate occurrence of elements with different characteristics in the DON enterprise namespace.[7] This reduces the level of effort to analyze, map, and transform elements that are unique to a particular functional area.

A further advantage of the global approach is that it focuses on *both* the individual global data elements (property and representation terms) and object level information (object classes defined as global complex types). This provides the DON a standardized, semantically unambiguous naming convention and a granular data management capability.

[7] A global element with the same semantics as another element that is already registered in the registry will not be duplicated.

The DON approach recognizes that not all leaf node (BBIE) elements can be globally named.[8] Certain leaf node (BBIE) elements can be defined as local.[9] Under certain circumstances, DON developers have the flexibility to create local elements for leaf node (BBIE) element of types "Identifier," "Code," and "Measure." However, these local elements must come from an enterprise-level controlled vocabulary, based on DON-approved standards that will be instantiated in the DON XML registry. The proposed process will place the responsibility on the FNC and BSC to identify when a local element should represent an identifier, code, and measure.

As highlighted in the example below, local elements were created for leaf node (BBIE) elements deriving from the UDTs of identifier, code, and measure.

Example: *Local elements identifier, code and measure (Example shown does not include all required documentation for the purposes of this illustration.)*

```
<!—Global Complex Type—>

<xsd:complexType name="TargetType">

    <xsd:sequence>
        <xsd:element name="TargetID" type="TargetIDType"/>
        <xsd:element ref="bie1-0:Description"/>
        <xsd:element ref="bie1-0:AssignedOrdnanceREF"/>
        <xsd:element name="TargetCountryCode" type="TargetCountryCodeType"/>
        <xsd:element name="SurfaceTemperatureMeasure"
        type="SurfaceTemperatureMeasureType"/>
    </xsd:sequence>
</xsd:complexType>
```

3.3 Conclusions

The DON approach meets current W3C Schema specifications. The approach can be summarized as follows:

- Uses globally defined semantics for all data elements.
- Generally follows emerging voluntary consensus standards naming and design rules.
- Has maximum reusability. Both elements and types are reusable.
- Allows for moderate developer flexibility for limited use of FNC- and BSC-approved local elements.
- Allows flexibility in the use of Xpath expressions.

[8] That is, in some cases, the combination of property term and representation term may not be sufficiently unambiguous to merit a globally applicable definition.

[9] Both the FNC and BSC must approve which IDs, codes, and measures are used as local elements.

The DON selected the mixed element approach as the best long-term choice for enterprise interoperability. This mixed data management approach has the following key advantages:

- Promotes enterprise standardization and maximum reusability. The majority of data elements have globally defined semantics that are globally defined, stored, and reused.
- Adheres to the concept of authoritative source data.
- Adheres to emerging standards, including the OASIS UBL Naming and Design Rules[10] and the UN/CEFACT electronic business XML (ebXML) Core Components Technical Specification (CCTS), and supports the ISO 11179 naming conventions.
- Allows for shorter tag names and reduced redundancy.
- Provides a universal semantically unambiguous data element framework as the foundation for an enterprise XML registry/repository.
- Meets a key DON XML metadata registry and repository requirement that *all* XML data elements be registered in the enterprise DON XML registry/repository.
- Provides the flexibility to use local elements where appropriate, namely, for identifiers, codes, and measures.

The mixed element approach provides the DON with the best solution to achieve the desired level of enterprise-level management, standardization, and interoperability. The mixed element approach also ensures that developers are given sufficient guidance and flexibility to rapidly develop XML in a manner consistent with departmental enterprise requirements.

[10] With the addition of the local elements in accordance with the mixed element approach presented in this paper.

Section 4
Namespace and Schema Modularity Rules

The DON enterprise architecture must be able to support multiple XML namespaces and a robust XML schema modularity architecture that allow for successful object and schema versioning. The DON enterprise architecture must conform to higher-level DoD architecture frameworks, guidance, and specifications. In addition, the DON standard for XML namespaces will continue to interoperate with the DoD registry. The interoperability requirements for a large amount of diverse enterprise XML data can quickly present a management problem, without effective standards to define a common XML namespace and schema modularity framework. Thus, a DON standard for XML namespace and schema modularity is critical to the success of an XML repository and any future interdepartmental processing of such data.

4.1 Purpose

The purpose of this section is to summarize the DON XML namespace and schema modularity approach. The BSC approved the namespace and schema modularity approach after significant research, analysis, and discussion.

4.2 XML Namespace Rules

DON XML Naming and Design Rules, Version 2.0 has adopted the Uniform Resource Name (URN) as the DON standard for Uniform Resource Identifiers (URIs). This namespace identifier must be a URI reference that conforms to the Internet Engineering Task Force (IETF) request for comments (RFC) 2396, *Uniform Resource Identifiers: Generic Syntax*.[1]

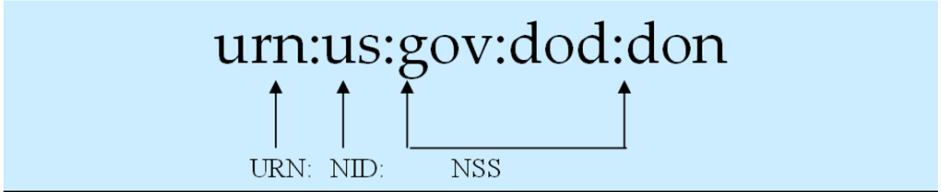
The DON standard for URIs is the URN. URNs are designed as persistent names that, once published, will not change, a requirement for an effective schema namespace. URNs can be registered in a global namespace identifier directory, providing the same opportunity as a URL to be resolvable and store information pertinent to the schema. In addition, the SchemaLocation attribute may be used to provide information on where the schema resides, rather than trying to use a namespace to identify a storage location.

[1] T. Berners-Lee, R. Fielding, and L. Masinter; Internet Engineering Task Force (IETF) RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, Internet Society, August 1998.

The concept of XML namespaces is defined in the W3C XML Namespaces technical specification. XML namespace features are published in the W3C XML Schema (XSD). The DON NDR recommends using XSD, and our discussions focus on the use of namespaces in XSD.[2]

Namespace Rule	
[NMS1]	All DON enterprise and development namespaces MUST use the base URN "urn:us:gov:dod:don."

The DON XML namespace structure follows the URN structure defined by the IETF Network Working Group in RFC 2141 and is in alignment with the federal government namespace recommendation.[3] That structure contains the URI consisting of the URN, the namespace identifier (NID), and namespace-specific string (NSS).[4] Following is an example:



The NID is represented by the string "US." The NSS conforms to the second- and/or third-level domain as registered with the domain and registration services for DoD.

The URN "urn:us:gov:dod:don" is the "base URN" for all DON XML namespace declarations; further hierarchical segmenting of the URN is allowed for both enterprise and development schema; this topic is detailed later in this section.

To illustrate how the naming convention would be applied, we use the available XML URN for DoD, which is "urn:us:gov:dod," as shown in the following examples:

- urn:us:gov:dod:don
- urn:us:gov:dod:don:navy

[2] As defined in RFC 2396, a URI is a "compact string of characters for identifying an abstract or physical resource." A URI scheme can be "a locator, a name, or both." A URI locator scheme is in the form of a URL, and a URI name scheme is in the form of a URN. URLs generally define a location but are not required to be a resolvable Internet or World Wide Web address. URNs are required to provide a globally unique and persistent reference even if the URL subset of the URI scheme ceases to exist.

[3] LMI, *Recommended XML Namespace for Government Organizations*, GS301L1, Jessica L. Glace and Mark R. Crawford, August 2003.

[4] R. Moats, Internet Engineering Task Force (IETF) RFC 2141, *URN Syntax*, Internet Society, May 1997.

- urn:us:gov:dod:don:usmc
- urn:us:gov:dod:doa.

The convention for DON URNs will be as follows:

Namespace Rule	
[NMS2]	URNs MUST be in lowercase, except multiple words, which MUST use lower camel case and the root element, which will use upper camel case.

The DON is responsible for managing the delineation of the namespace past the second-level registered domain, as would other DON components that want to further extend their URNs.

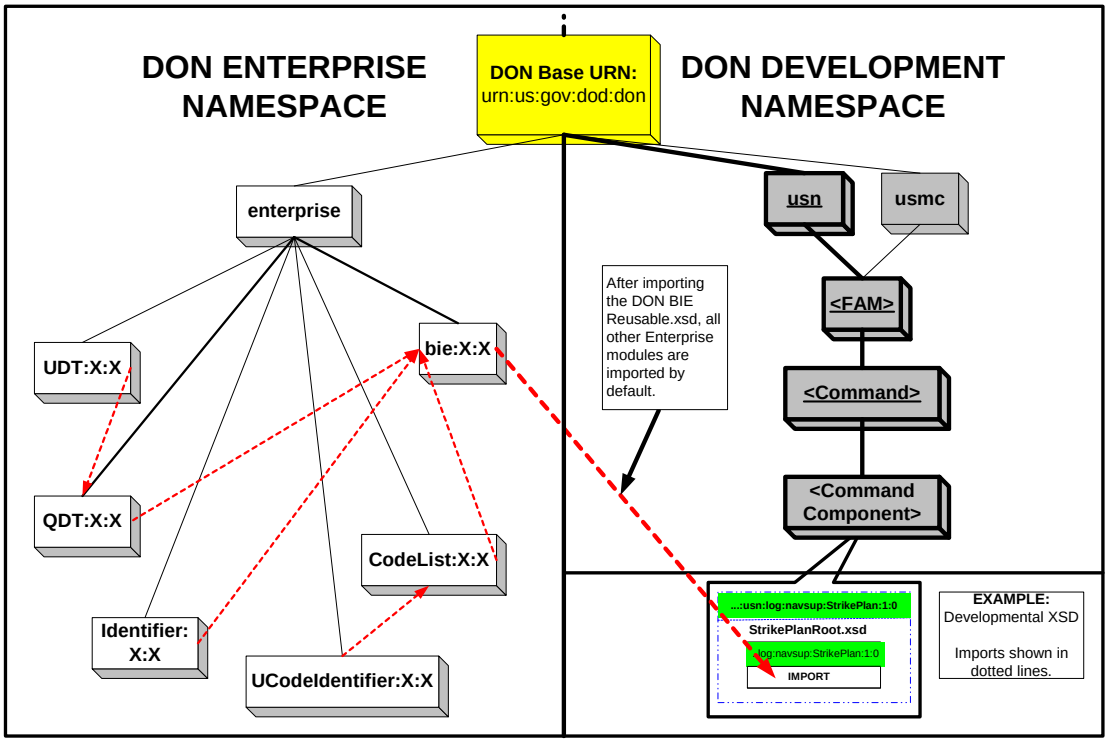
Namespace Rule	
[NMS3]	All Business Standards Council (BSC) approved enterprise (root) schema MUST reside in the DON enterprise root namespace.

The DON schema namespaces have two major classes:

- Enterprise schema namespace—XSD files that have been harmonized into the DON enterprise namespace and therefore have a target namespace URN that begins "urn:us:gov:dod:don:enterprise."
- Development schema modules—XSD files developed by departments within the DON that have not been harmonized by the FNCs. These will begin with the DON base URN and will include service branch, functional area, department hierarchy, component types (BIE, code, etc.), and version information.

Figure 4-1 details the allowable URN extensions for both enterprise and development namespaces.

Figure 4-1. DON URN Hierarchy for Enterprise and Development Namespaces



The right half of Figure 4-1 illustrates a proposed convention for departments to adopt in creating development schema modules (discussed in another section of this paper). The adherence to the enterprise URN convention is mandatory for all enterprise XML components, while the use of the development URN convention is recommended but not required.

Namespace Rule

[NMS4] The namespace for DON root schema holding enterprise status MUST be of this form:

urn:us:gov:dod:don:enterprise:schema:<root name>:<major>:<minor>.

To meet the BSC's requirements of having a single enterprise namespace, the following convention was adopted:[5]

Example: *DON Enterprise URN Convention*

urn:us:gov:dod:don:enterprise*

Base URN

DON Enterprise URN

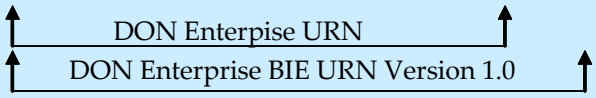
* The term "enterprise" is appended to the base URN.

[5] Here the term "namespace" refers to the establishment of a single logical space for collection of enterprise components; this definition is similar to the one used in the DoD registry.

The enterprise XML namespace established by this URN is further segmented to provide for management and control of various types of enterprise components. The DON Enterprise BIE Reusable XSD is assigned to the XML namespace URN created by appending the name of the "bie" and version data to the enterprise namespace, as shown below:

Example: *Base URN for the DON Enterprise BIE URN*

urn:us:gov:dod:don:enterprise:bie:1:0



Accordingly, following the URN hierarchy described in Figure 4-1, the following rules describe the required enterprise namespace conventions.

Qualified and Unqualified Datatypes Schema modules each have their own namespace within the DON domain.

Namespace Rules	
[NMS5]	The namespace for DON QDT schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:qdt: <major>:<minor>.
[NMS6]	The namespace for DON UDT schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:udt: <major>:<minor>.

Code lists and identifiers each have their own namespace within the DON domain.

Namespace Rules	
[NMS7]	The namespace for DON Code List schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:<codeListname>:<major>:<minor>.
[NMS8]	The namespace for DON Identifier schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:<identifierName>:<major>:<minor>.
[NMS9]	The namespace for DON Unqualified Code List and Identifier schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:UCodeIdentifer: <major>:<minor>.
[NMS10]	The namespace for DON schema holding development status MUST be of this form: urn:us:gov:dod:don:<service branch>:<FAM>:<command>:<component>:<root name>:<major>:<minor>

Development and enterprise namespaces have been created to allow the DON departments the flexibility to build XML schema to meet their immediate needs. At the same time, the FNCs will be attempting to harmonize these locally developed XML components to produce enterprise BIEs that are reusable by any department regardless of functional area. After BSC review and approval, the schema that was submitted will be propagated to the DON enterprise namespace.

The following are examples of development URNs used as target namespaces of Development Schema modules:

Example: *Developmental URN*

urn:us:gov:dod:don:usn:log:navsup:navsisa:StockCheck:1:0*

DON Base URN

NAVSISA* Stock Check Message Set Version 1.0

*NAVSISA is a component of the Navy Supply Center (NAVSUP), so the department made the decision to further segment its development URN to indicate that the URN is for components developed by NAVSISA.

urn:us:gov:dod:don:usn:ccc:spawar:sscsd:2833:bie:1:0

DON Base URN

Developmental URN for SPAWAR* BIE Components Version 1.0

* In this case the BIE components are being developed in the C3 FAM for a command and control application by SPAWAR System's Center San Diego, Code 2833. The hierarchical structure convention "spawar:sscsd:2833" is determined by SPAWAR. It is possible that other development URNs may be created at SPAWAR by other SPAWAR component activities in other functional areas.

Namespace Rule

[NMS11] All DON namespace declarations MUST be qualified.

Schema objects are associated with a namespace identifier through a user-defined namespace prefix, making the constructs "namespace qualified." In the following example, the namespace identifier is "urn:us:gov:dod:don" and the namespace prefix is "don":

Example: *DON Namespace Prefix*

```
<xsd:Schema xmlns:don="urn:us:gov:dod:don">
```

This means that any construct in the schema with a namespace prefix of "don" belongs to the DON namespace, as in the following example:

Example: *DON National Stock Number Identifier*

```
<xsd:element name="don:NationalStockNumberIdentifier" type="bie1-
    0:NationalStockNumberType"/>
```

Namespaces allow XML elements with the same name to be used in the same schema with no adverse effects. In the following example, two "State" elements are used in the same schema, but they are associated with two different namespaces. One element represents a U.S. state abbreviation (AK, AL, AR) in the DON's namespace, while the other represents the state of water quality (acidic, basic, high turbidity) in Department of Army's namespace:

Example: *Namespace Prefix Association*

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:Schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
        xmlns:don="urn:us:gov:dod:don" xmlns:doa="urn:us:gov:dod:doa">
    <xsd:element name="don:State" type="don:StatePostalCodeType"/>
    <xsd:element name="doa:State" type="doa:WaterQualityIndicatorType"/>
</xsd:Schema>
```

If the "State" elements described above were not in separate namespaces, an XML processor would generate an error. This condition is known as "name collision."

Namespace Rule

[NMS12] DON enterprise schema MUST declare a namespace using the xsd:targetNamespace attribute.

A target namespace is a "collector" of BIEs. A schema may have more than one declared namespace, but only one designated target namespace. A target namespace is declared using the namespace identifier of the selected namespace. In this example, the "`urn:us:gov:dod:don`" namespace is declared as the target namespace:

Example: *Target Namespace*

```
<?xml version="1.0" encoding="UTF-8"?>
        <xsd:Schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
        xmlns:don="urn:us:gov:dod:don"targetNamespace="urn:us:gov:dod:don">
```

Therefore, any element, attribute, or data type declared in this schema belongs to the schema's target namespace.

4.3 Schema Structure Modularity Rules

4.3.1 Enterprise Namespace

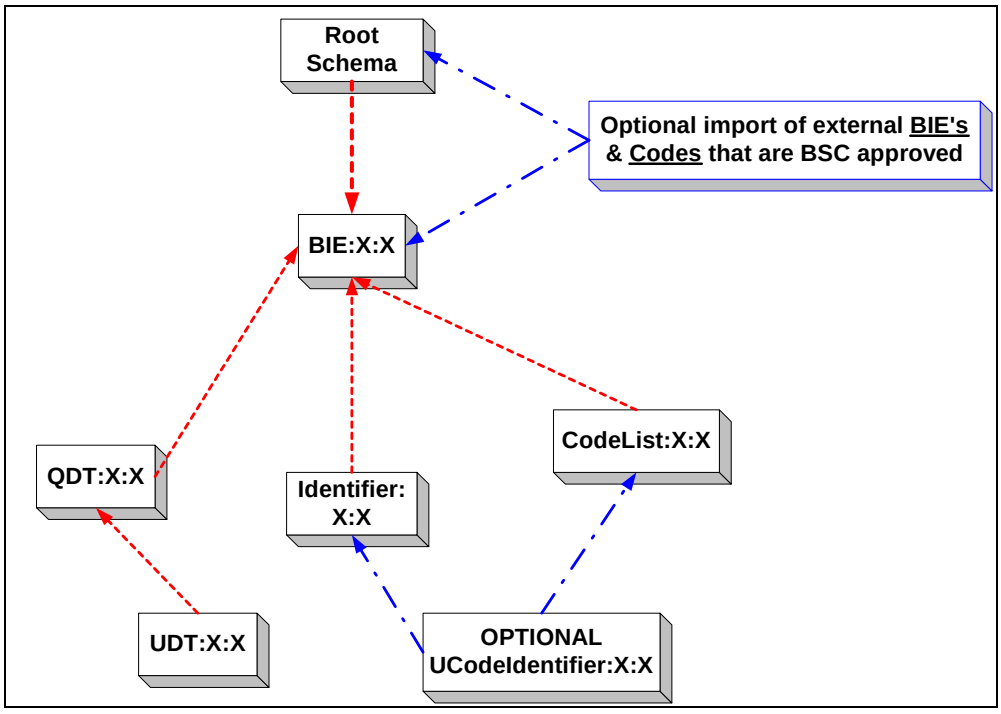
Namespace Rule	
[NMS13]	The DON enterprise namespace MUST contain only DON-developed schema modules.

Modularity describes the logical schema architecture that includes all XML BIEs. Schema modules that make up the DON enterprise namespace and modularity architecture include the following:

- Root schemas (business process' or documents)
- Qualified Code List and Identifier Schema modules
- Qualified and Unqualified Data Types modules
- DON BIE Reusable Schema module.

In many cases, however, it may be required to use schema components from an external authoritative data source, such as other DoD core components or from the international standards community. As shown in Figure 4-2, the modularity architecture allows for import of external schema, but all external components (core components, data types, code lists, etc.) must be approved by both the BSC and cognizant FNC or higher level DoD authority.

Figure 4-2. DON Enterprise Schema Modularity Architecture

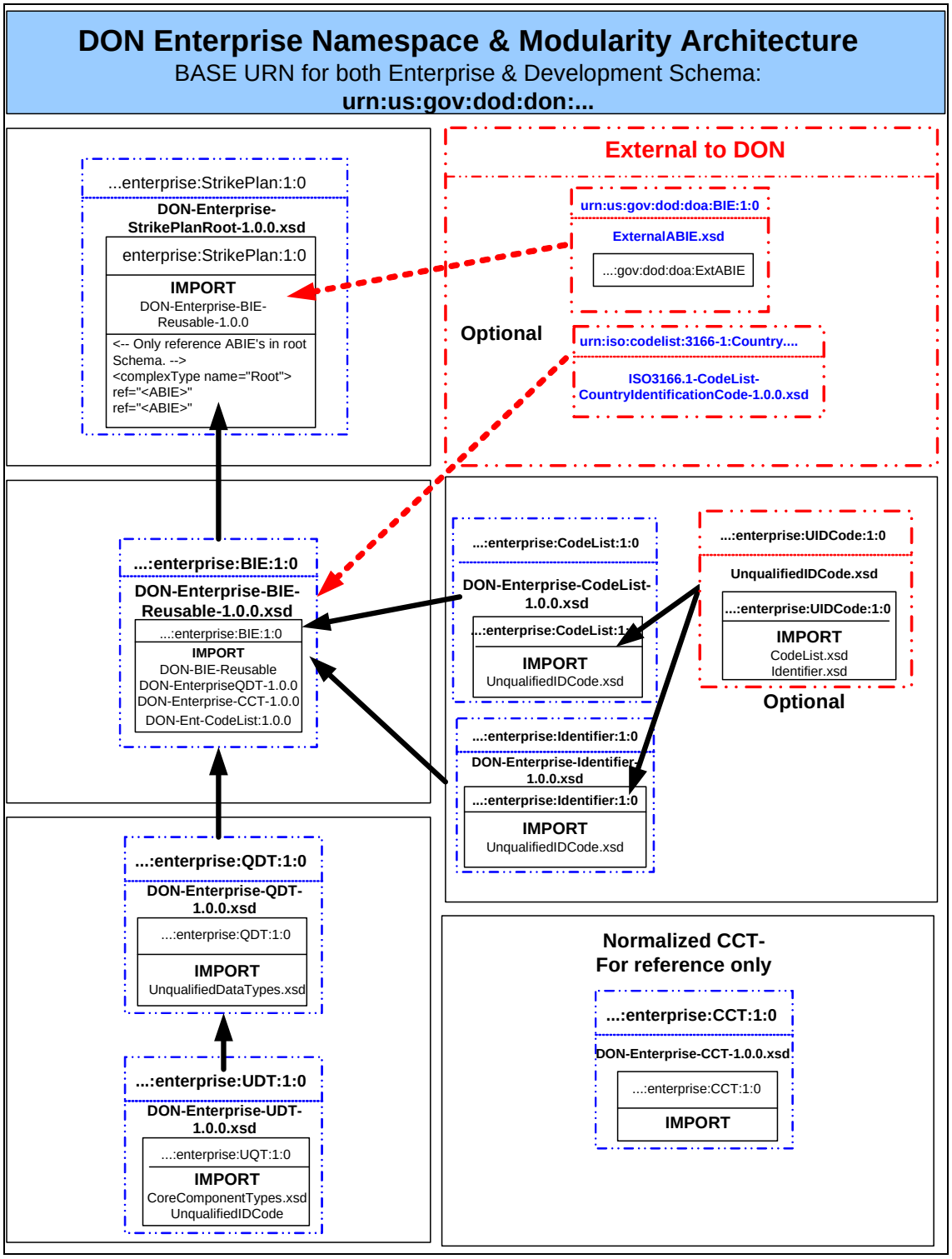


Schema Structure Modularity Rule	
[SSM1]	Enterprise level root schema MAY import external qualified data types, core components, code list, and identifier schema that have been approved by the BSC and cognizant FNC authority.

The Enterprise Root Schema modules will reside in the Enterprise Root Schema namespace (don:ent:strike:1:0), the Qualified and Unqualified Data Types modules will reside in their own discrete namespace (don:ent:qdt:1:0 & don:ent:udt:1:0), the Unqualified and Qualified Identifier and Code List schema modules will reside in their own namespaces (don:ent:id:1:0 and don:ent:code:1:0) and the BIE Reusable Schema module will reside in the enterprise BIE namespace (e.g., don:ent:bie:1:0). Figure 4-3 illustrates this architecture.

Schema Structure Modularity Rule	
[SSM2]	Every DON enterprise root schema MUST import the DON Enterprise Reusable BIE schema (DON-Enterprise-Reusable-z.z.z.xsd).
[SSM3]	All enterprise global elements and named complex types representing ccts:Components must reside in the Enterprise BIE Reusable Schema module.

Figure 4-3. DON Schema Modularity—Enterprise Namespace



4.3.2 Schema Module Types

As harmonization is conducted by FNCs and the BSC and enterprise XML BIEs are produced, DON departments can replace locally developed BIEs with approved

enterprise schema BIEs. Being able to quickly evolve existing schemas with FNC-harmonized XML components is critical to achieving enterprise interoperability with XML.

The different types of schema modules included in both enterprise and development document schemas include the following:

- *Root Schema Module.* The Root Schema module is the control schema for all other schema modules specific to an XML transaction. The Root Schema imports the BIE Reusables, Qualified and Unqualified Data Types, and Code List and Identifier Schema modules.
- *Core Component Types Schema (CCT) Module.* The DON will reuse the OASIS UBL XML Core Component Schema. This is required to facilitate maximum reusability across the DON. The CCT contains the foundation for all DON data types. The UDT is a further extension of these core data types and both the DON Enterprise and Development Root Schema modules will import the UDT. Note: The CCT Schema module will *not* be imported into the DON UDT and is included to represent the normalized reference of the fundamental CCTS types.
- *Unqualified Data Type (UDT) Module.* The DON UDT module provides the basic data types (complexTypes) from which all other data types must derive. The UDT is reusable external to the DON.
- *Qualified Data Type (QDT) Module.* The DON QDT module provides all qualified data types (complexTypes) that are derived from the UDT. It also includes their extensions, restrictions, and other domain constraints, such as facets. The QDT is reusable only within the DON.
- *BIE Reusable Schema Module.* All DON BIEs reside in the BIE Reusable Schema module. The BIE forms the central unit of XML object reusability and attempts to maximize reusability and interoperability across the DON.

  XML schema modules containing the XML BIEs are developed as Development BIE Schema modules. These development modules are then submitted to the cognizant FNC and BSC for approval.

- *Code List and Identifier Schema Modules.* Codes and identifiers are used to represent an exact set of business semantics or provide a unique identifier. It is possible to construct an XML schema module that provides an allowable set of enumerated codes or identifiers as possible valid values of an XML element representing a code or identifier. Enterprise Code List and Identifier modules will allow departments responsible for maintaining code lists and identifiers to develop and maintain all approved code lists and identifiers.

Code lists reside in their own namespace. See the namespace rules for further details.

Namespace Rule	
[NMS14]	Each DON:CodeList schema module MUST be maintained in a separate namespace.

FNCs will be responsible for identifying and approving code lists and identifiers in conjunction with the BSC. FNCs will provide locally developed Code List and Identifier modules as inputs. The standard for DON Code List and Identifier modules is based on the recommendations of the OASIS UBL NDR subcommittee. Therefore, code lists developed according to these standards will be reusable in all DON enterprise XML schemas.

4.3.3 Development Namespace

The DON development namespace provides the schema developer with a virtual workspace to create and store schema. Figure 4-4 details the modularity architecture of the normative submission in the DON development namespace. The developer must submit a "normative" schema to both the DoD registry and cognizant FNC. The FNC will be responsible for submitting the normative schema for BSC approval.

Schema Structure Modularity Rule	
[SSM4]	All DON development root schemas MUST be submitted to both the cognizant FNC and the appropriate namespace in the DoD registry.

The DON rules require a normative schema submission. At a minimum, the normative schema must contain

- an enterprise root schema and
- a separate BIE Schema module with *no* namespace declaration. (This is typically labeled a "no-namespace" submission.)

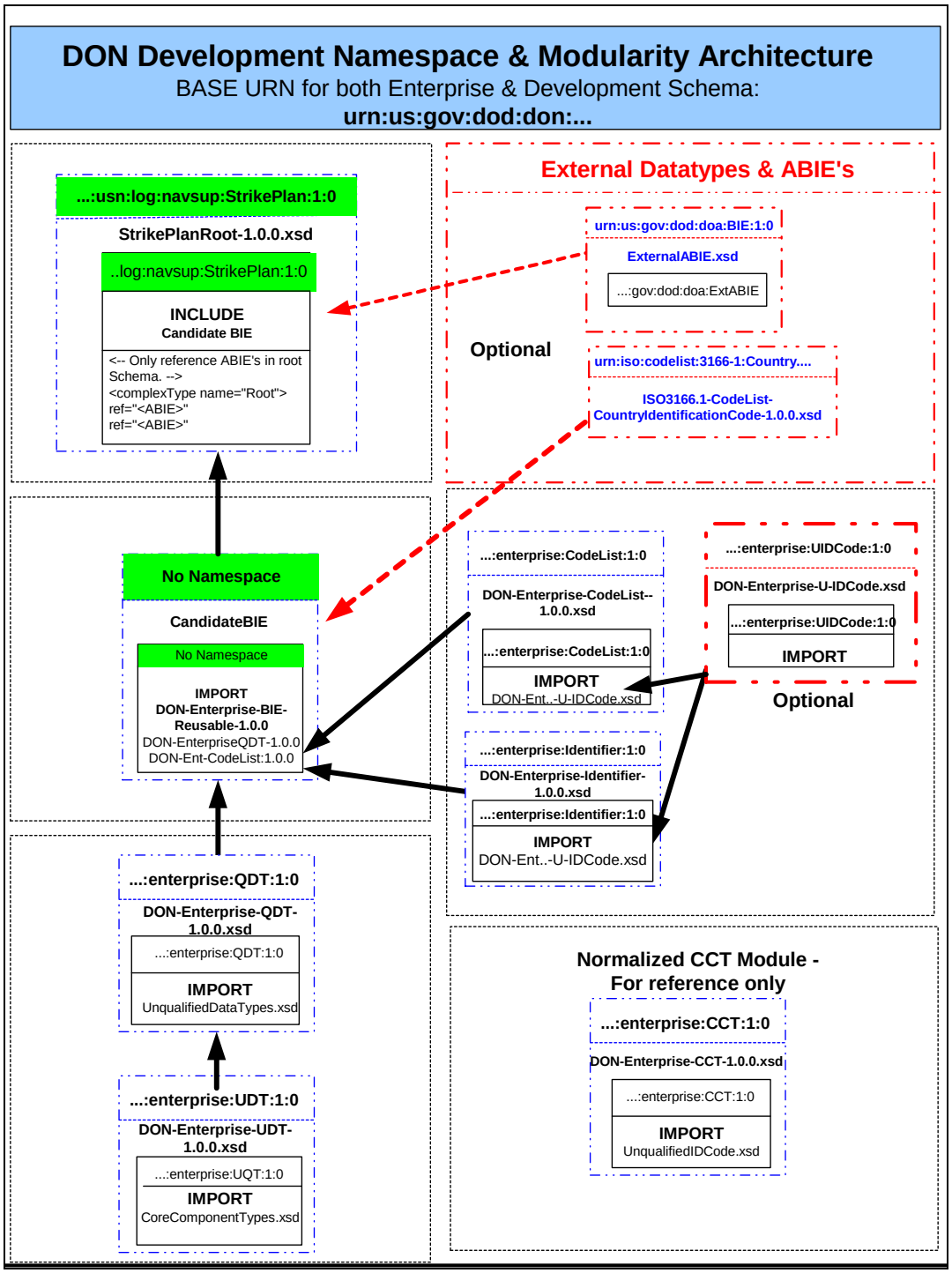
Schema Structure Modularity Rule	
[SSM5]	Development global elements and named complex types MUST reside in a no namespace schema that is included in the root schema.

The DON_BIE_Reusable.xsd module contains all DON approved BIEs. The concept of BIEs was adopted from the CCTS and the UBL as a means to express reusable information entities and bind those to XML schema complex types and global elements.[6]

Schema Structure Modularity Rule	
[SSM6]	Development qualified data types (complex types) MUST reside in a no namespace qualified data types schema that is included in the root schema submission.

[6] Managing enterprise-level BIEs will be a critical success factor to DON interoperability.

Figure 4-4. DON Schema Modularity—Development Namespace



4.3.4 Run-Time Schema

The developer has the option of creating a "run-time" schema, where BIEs specific to that business process are localized into a single "flat" schema for performance improvement to production applications. The run-time schema does *not* qualify as the normative entry required by the DON registry.

Schema Structure Modularity Rule	
[SSM7]	A run-time schema MAY be created to meet performance requirements of the application in the run-time environment.

All modifications, updates, revisions and new releases must first go through the DON schema approval process before the changes are incorporated into the run-time schema.

Schema Structure Modularity Rule	
[SSM8]	All modifications, updates, revisions, and new releases MUST first go through the DON schema approval process before the changes can be incorporated into the run-time schema.

In most cases, this requirement supports a DON department's existing configuration management processes. For example, assuming an approved schema has been translated into a run-time schema and is running in production, the following steps are required before making a change to the production application:

1. The organization submits a normative schema, which includes all new changes, to the cognizant FNC.
2. The FNC submits the change to the BSC for approval.
3. The BSC approves, harmonizes, and integrates the changes into the DON enterprise namespace.
4. After receiving approval, the submitting organization incorporates the change into its existing production application (schema). (Typically, before changing a production application, the owners of the application will regression test the change in an appropriate development, testing, and staging environment.)

Roles and responsibilities for the DON schema submission process are further detailed in the *DON BSC Operating Procedures.*

Schema Structure Modularity Rule	
[SSM9]	Xsd:Include MAY only be used in a Development or enterprise run-time root schema.

Xsd:Include MAY only be used in a development or enterprise run-time root schema, for the following reasons:

- To meet performance requirements of the application in the run-time environment
- To meet development requirements that are specific to the local organization.

Section 5
Versioning Rules

DON schema components must be version controlled to address potential backward-compatibility issues inherent in XML processing. XML namespace and schema versioning provide the DON with the best long-term versioning strategy because it allows the DON developer to access both the latest enterprise core components without interrupting production applications that reference an older component.

To make this strategy successful, version information must be contained in both in the schema and in the schema's target namespace. This section will detail the DON versioning rules for both namespace and schema.

The initial release of DON enterprise reusable components will be to a 1.0 namespace (the URN ends in "1:0"). Subsequent new components are added to the 1.0 namespace by creating a new XSD in the namespace. The previous XSD is included in the new XSD so that all 1.0 components are available from it.

The DON process for identifying and harmonizing schema components into the enterprise means that new components will need to be added to the 1.0 enterprise namespace at intervals and new versions of existing components will need to be added to higher versioned namespaces. This addition of new components to a namespace that already has components is accomplished by creating a new schema in the namespace. This new schema then includes the previous schema.

5.1 Purpose

The purpose of this section is to summarize the DON XML versioning rules and approach.

5.2 Namespace (URN) Versioning Rules

Version information is included in the namespace (URN). Namespace versions are represented by a major and minor version number, with an optional revision identifier for draft schema in the DON development namespaces. For example, the namespace URN for the draft invoice domain has this form:

Example: *Sample - Invoice URN*

…:don:enterprise:schema:invoice:<major>:<minor>

The component version is bound to the namespace version; this binding allows namespace aware XML processors to differentiate component versions through the use of

XML namespace and prefixes. This capability makes it possible to achieve backward compatibility as components are added, deleted, or modified.

Versioning Rule	
[VER1]	XML schema version information MUST be defined in a schema and match the version of the namespace in which it resides.

Example: *Synchronizing URN and Schema Versions*

URN Version: urn:us:gov:dod:don:enterprise:StrikePlan:**2:1**
XSD version: <schema version="**2.1.**1">

Versioning Rule	
[VER2]	The major-version field MUST equal "1" for the first release of a namespace.

For example, the first namespace URN for the first major release of the StrikePlan has the form:

Example: *StrikePlan 1.0 URN*

urn:us:gov:dod:don:enterprise:StrikePlan:1:0

The second major release will have a URN of the form:

urn:us:gov:dod:don:enterprise:StrikePlan:2:0

Versioning Rule	
[VER3]	The major-version field of a namespace or schema MUST be incremented when the proposed Schema changes impact the compatibility of any previous XML instance based on the related Schema.

If a change does not break compatibility, then only the minor version need change. Subsequent minor releases begin with *minor-version* 1.

Versioning Rule	
[VER4]	The minor-version field of a namespace or schema MUST be incremented if all XML instances will continue to validate successfully with the new version of the schema.

An optional <revision> identifier can be appended to the URN when the schema is in the DON *development* namespace. When the schema is approved, removal of the <revision> identifier is required before promoting the schema into the DON enterprise namespace.

Example: *Revision Identifier*

urn:us:gov:dod:don:navsup:navsisa:StrikePlan:1:1:**a**

All schema will have the major minor versioning in the namespace.

Versioning Rule	
[VER5]	Every major- or minor-version enterprise DON schema MUST include the version information of: <major>:<minor> of the namespace name.

Example: *URN Versioning*

urn:us:gov:dod:don:enterprise:StrikePlan:1:1

Minor version releases are created by using incremental version numbers so that the number is no longer equal to zero.

5.3 Schema Versioning Rules

A version number that includes the namespace major and minor version and an additional identifier to capture the sequential schema revision must be included in schema that reside in the DON enterprise BIE or QDT namespace. Each time a developer modifies a schema and subsequently adds the revised schema to the namespace, the <schema version number> must be incremented.

Versioning Rule	
[VER6]	The first schema module to appear in a DON Enterprise BIE or Qualified Data Type namespace MUST have a version attribute equal to the following: <major-number>:<minor-number>:<schema version number>.

The schema version number has three positions; the first two map to the namespace version, and the third is a sequential number that indicates the number of the schema with respect to the total number of versions of the schemas in the namespace.

Versioning Rule	
[VER7]	Schema in the DON Enterprise BIE or Qualified Data Type namespaces MUST increment the third digit of the schema version attribute when a schema is changed.

Example: *Schema Version*

<xsd:schema targetNamespace="…:bie:1:0" **version="1.0.3"**>

This example indicates that the schema is the third in the 1.0 namespace (the first was 1.0.1).

The purpose of maintaining both namespace version and schema version is to provide a mechanism to provide better configuration management within a particular namespace. The original or previous version of the schema will continue to reside in the namespace. This ensures that a developer, when adding a new schema in the same target namespace, is required to import only the target namespace and specify the location of the most recent schema in the namespace.

<xsd:import namespace="…:bie:1:0" schemaLocation="…/DON_BIE_Reusable.xsd"

5.3.1 Enterprise Reusable Schema

To ensure that development organizations are using the latest version of components, developers should import only the most recent schema file in a namespace. Since multiple schema will exist in a single enterprise reusable namespace, root schema developers will have a choice as to which schema they reference when importing a namespace. Under most circumstances, developers should reference the latest schema to ensure that any new components are available.

Minor versions must maintain backward compatibility with instances based on previous minor versions. When minor versioning a root schema, the previous minor version is imported.

Versioning Rule	
[VER8]	Minor version changes to enterprise, qualified data types, and root schema components MUST be backward compatible and validate all previous versions of XML instances.

Versioning is separated into major and minor to enable better interoperability by maximizing backward compatibility. As such, when a version change enables instances based on previous component versions to be validated by a schema employing a new version, the change is said to be minor. Any other version change must result in a major version.

Since minor versions are required to maintain backward compatibility, the changes to a root schema that can be considered minor are limited to extension of the schema with new elements (either development or enterprise). The components existing in the previous minor version root schema must remain intact and therefore are included into the new version.

Versioning Rule	
[VER9]	New versions of the schema created in the DON Enterprise BIE or Qualified Data Types namespace MUST *include* the previous schema version.

Example: Include - xsd:include

```
<xsd:include schemaLocation="…/BIE/DON_BIE_Reusables.xsd"
```

5.3.2 Core Component Versioning Rules

CCTS component definitions and its CCTS dictionary entry name cannot be changed without altering the business meaning of the core component. Changing a component's business meaning fundamentally creates a new component. If the fundamental business meaning of a core component must be changed, a new core component should be created, rather than versioning an existing component.

Versioning Rule	
[VER10]	New core component versions MUST not alter the component's ccts:DictionaryEntryName or definition.

5.4 Versioning Summary

The DON versioning strategy successfully addresses any future backward-compatibility issues inherent in XML processing. XML namespace and schema versioning provide the DON with the best long-term strategy because it allows the developer to access the latest DON enterprise components without interrupting production applications (that reference an older component).

The DON namespace and modularity architecture supports the overall versioning strategy by creating a DON enterprise namespace hierarchy that all enterprise root schemas must follow. Accordingly, the DON versioning rules provide a further means to maintain configuration control of the DON enterprise schemas.

Section 6
General XML Rules

The DON standard for XML schema specification is the W3C XSD. The W3C Schema specification has become the generally accepted schema language that is experiencing the most widespread adoption. Although other schema languages exist that have their own advantages and disadvantages, the DON has determined that the best approach for developing an enterprise XML standard is to base its work on W3C XSD.

Standards Adherence Rule	
[STA1]	All DON XSD schema design MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

A W3C technical specification holding recommended status represents consensus within the W3C and has the Director's stamp of approval. Recommendations are appropriate for widespread deployment and promote W3C's mission. Before the Director approves a recommendation, it must show an alignment with the W3C architecture. By aligning with W3C specifications holding recommended status, the DON can ensure that its products and deliverables are well suited for use by the widest possible audience with the best availability of common support tools.

Standards Adherence Rule	
[STA2]	All DON schema and messages MUST be based on the W3C suite of technical specifications holding recommendation status.

Each section of this guide contains rules that are applicable to a specific facet of the XML technology. Some design rules transcend individual facets, however, and are more appropriate as general XML rules that apply to all XML development efforts. This section contains those high-level XML design rules.

6.1 Purpose

The purpose of this section is to summarize the DON general XML rules and approach for general XML development.

6.2 Overall Schema Structure

Per XML 1.0, there is exactly one element, called the root, or document element, no part of which appears in the content of any other element. XML 1.0 further states that the root element of any document is considered to have signaled no intentions as regards application space handling, unless it provides a value for this attribute or the attribute is declared with a default value. W3C XSD allows for any globally declared element to be the document root element. To keep consistency in the instance documents and to adhere

to the underlying process model that supports each DON schema, it is desirable to have one and only one element function as the root element. Because the DON follows a global element declaration scheme (see Rule ELD1), each DON schema will identify one element declaration in each schema as the document root element. This will be accomplished through an xsd:annotation child element for that element in accordance with the following rule:

Root Element Declaration Rule	
[RED1]	Each DON Root-Level Schema module MUST identify at least one global element declaration that defines the content in the schema expression. That global element MUST include an xsd:annotation child element, which MUST further contain an xsd:documentation child element that declares the following: "This element MUST be conveyed as the root element in any instance document based on this schema expression."

Example: *Root Element—StrikePlan*

```
<xsd:element name="StrikePlan" type="StrikePlanType"/>
    <xsd:complexType name="StrikePlanType">
        <xsd:annotation>
            <xsd:documentation>
                <doc:ComponentType>Root</doc:ComponentType>
                    <doc:Description>This element MUST be conveyed as the root
                    element in any instance document based on this schema
                    expression.</doc:Description>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="Strike" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
```

The name of the root element must conform to the purpose of the business process or document.

Root Element Declaration Rule	
[RED2]	Every root element in a DON document MUST be named according to the portion of the business process or document it initiates.

6.3 General XSD Rules

The features of W3C XML Schema allow for flexibility of use for many different and varied types of implementation. The DON NDR uses the following rules to allow for a more consistent use of these features:

General XSD Rule	
[GXS1]	The root element in all DON Schema modules MUST contain the following declaration: "xmlns:xsd=http://www.w3.org/2001/XMLSchema."

To avoid overloading implementation systems with unnecessary documentation, developers have an option to create a schema without the documentation. This schema is for run-time and must be an exact mirror of the documented version.

This run-time schema may be flattened; that is, it may contain all other XML schema components, including those that are defined in other imported schema modules.

General XSD Rule	
[GXS2]	DON schema developers MAY provide a run-time schema devoid of documentation in addition to the fully annotated version.

6.3.1 Built-in Simple Types

There are 44 simple types built into XML Schema. They are specified in Part 2 of the XML Schema Recommendation. These built-in types should be used in the designing of schema. Simple types are the concrete representations of the datatypes of abstract concepts such as "integer."

The DON Enterprise UDT Schema module represents the XML Schema built-in types and fundamental CCTS types. The DON UDT Schema module declares the built-in types to be used.

General XSD Rule	
[GXS3]	DON UDT built-in simple types MUST be used wherever possible.

6.3.2 XSD:substitution

XSD:substitution is "a feature of W3C XML Schema, allowing you to define groups of elements that may be used interchangeably in instance documents. They are not declared as element groups, but through the substitutionGroup attribute of xsd:element global definitions."

The DON has made the decision to *not* allow the use of substitution groups due to the following issues:

- SubstitutionGroup "may work with some schema processors but relies on a liberal interpretation of the Recommendation, which may lead to interoperability issues."
- Both the extension of xsd:choice and the restriction of xsd:substitution could impact interoperability.

General XSD Rule	
[GXS4]	The xsd:substitution groups feature SHOULD NOT be used.

6.3.3 XSD:final

XSD:final is only allowed in the following rule, to stop further restriction or extension on complexTypes that are already derived from restriction.

General XSD Rule	
[GXS5]	The xsd:final attribute MUST be used on xsd:complexType definitions derived by restriction to prevent further restriction or extensions.

6.3.4 XSD:notation

XSD:notation is used to declare the format of non-XML data. A notation in XML is just like the notation declarations in DTDs. The main difference is that W3C XML Schema notations are namespace-aware and can be imported between schemas. When these declarations are used, the notations are used in xsd:enumeration facets to create simple types.

The notation datatype is used to declare links to external non-XML content (for example, image data) and then associate that content with an external application that handles it.

Notation is a built-in legacy simple type and are very seldom used in production applications, and not optimal for the DON work.

General XSD Rule	
[GXS6]	xsd:notations MUST NOT be used.

6.3.5 XSD:all

Used within a group, xsd:all has the same meaning as when it is used directly under xsd:complexType, except that there are no minOccurs and maxOccurs attributes and it cannot be marked as optional. The xsd:all compositor requires occurrence indicators of minOccurs=0 and maxOccurs=1. The xsd:all compositor allows for elements to occur in any order. The result is that in an instance document, elements can occur in any order, are always optional, and never occur more then once. Such restrictions are inconsistent with data-centric scenarios such as most of the work in the DON.

Another disadvantage of xsd:all is that it cannot be repeated any further. This limits the use of xsd:all to the first occurrence of its set of elements. If a content model requires an element that occurs more than once, then xsd:all cannot be used.

General XSD Rule	
[GXS7]	The xsd:all element MUST NOT be used.

6.3.6 XSD:choice

The choice groups allow any one of several child elements to appear in content. Groups are either all, choice, or sequence.

Choice allows for more complex content models. A choice group of element declarations is used to indicate that only one of the corresponding conforming elements must appear.

General XSD Rule	
[GXS8]	The xsd:choice element MAY be used.

6.3.7 XSD:any

General XSD Rule	
[GXS9]	The xsd:namespace attribute MUST be defined when using the xsd:any element and MUST have a URI value representing the namespace of a BSC-approved XML vocabulary.

6.3.8 XSD:nil

Null values must not be used. XSD:nil is used to differentiate between a zero-length strings, zero values and an undefined values. Elements that carry null values can cause XML processor errors because the database source may be expecting an integer as defined by the element type. To mitigate this type of error, the following rule is applied:

General XSD Rule	
[GXS10]	The xsd built-in nillable attribute MUST NOT be used.

6.3.9 XSD:appinfo

Documentation in the schema is done with annotations. These annotations contain human-readable information or can contain application information. The DON recommends that developers do not use the feature xsd:appinfo or put the application information within the schema.Appinfo should be used only to convey non-normative data. Thus the following rule:

General XSD	
[GXS11]	DON schema SHOULD NOT use xsd:appinfo. If used, xsd:appinfo MUST only be used to convey non-normative information.

6.4 Extension and Restriction

An existing DON type can be modified to fit the requirements of customization through XSD derivation. These modifications can include extension (adding new information to an existing type) or refinement (restricting the set of information allowed to a subset of that permitted by the existing type).

General XSD Rule	
[GXS12]	Complex type extension or restriction MAY be used.

6.4.1 XSD:complexTypes

It is assumed that, in many cases, in the development of new schema within the DON domain, there will be instances of reuse of schema that will need restriction to create the desired results.

General XSD Rule	
[GXS13]	Any xsd:complexTypes derived by restriction MUST NOT be further extended.

6.4.2 XSD:import

Import is used to bring schema modules into a master or root schema. The module resides in a different namespace; you then import it into the target namespace of the master or root schema.

General XSD Rule	
[GXS14]	The code list xsd:import element MUST contain the namespace and schema location attributes.

6.4.3 XSD:include

General XSD Rule	
[GXS15]	The xsd:include feature MUST be used only when applicable.

6.4.4 XSD:union

General XSD Rule	
[GXS16]	The xsd:union technique MAY be used for code lists.

Example: *xsd:union*

```
<xsd:simpleType name="">
    <xsd:union memberTypes=""/>
</xsd:simpleType>
```

6.5 Documentation

Documenting the XSD schema and XML instances of the DON is an important aspect of the success of XML across the DON. This documentation should enable new developers and users coming across this work the ability to understand what was done before and therefore keep consistency in the projects.

6.5.1 Annotation and Documentation

Annotation is an essential tool in understanding and reusing a schema. The DON, as an implementation of CCTS, requires an annotation to provide all necessary metadata required by the CCTS specification. Each construct declared or defined within the DON XML registry contains the requisite associated metadata to fully describe its nature and support the CCTS requirement. Accordingly, DON schema metadata for each construct is required, as delineated in the following sections.

6.5.2 Schema Annotation

Although the DON schema annotation is necessary, its volume results in a considerable increase in the size of the DON schema, with possible undesirable performance impacts. To address this issue, two schema may be developed for each DON schema. A fully annotated schema, normative submission, will be provided to facilitate greater understanding of the schema module and its components and to meet the CCTS and NDR metadata requirements. A schema devoid of annotation may also be created that can be used at run-time if required to meet processor resource constraints.

6.5.3 Embedded Documentation

The information about each DON BIE may be stored in spreadsheet format. DON spreadsheets can contain all necessary information to produce fully annotated schema. OASIS UBL has released a BIE spreadsheet that can be used as a template to store all BIE information. Fully annotated schema are valuable tools to implementers to assist in understanding the nuances of the information contained therein. DON annotations will consist of information currently required by Section 7 of the CCTS and supplemented by necessary information identified by the DON organization.

The absence of an optional annotation inside the structured set of annotations in the documentation element implies the use of the default value. For example, several annotations relate to context such as BusinessTermContext or IndustryContext whose absence implies that their value is "all contexts."

The following rules describe the documentation requirements for each data type definition.

Documentation Rules

[DOC1] Every data type definition MUST contain a structured set of annotations in the following sequence and pattern:

- UniqueIdentifier (mandatory): The identifier that references a data type instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. For example, BBIE, ABIE, ASBIE.
- DictionaryEntryName (mandatory): The official name of a data type.
- Definition (mandatory): The semantic meaning of a data type.
- Version (mandatory): An indication of the evolution over time of a data type instance.
- QualifierObjectClass (optional): The qualifier for the object class.
- ObjectClass: The object class represented by the data type.
- Qualifier Term (mandatory): A semantically meaningful name that differentiates the data type from its underlying UDT.
- Usage Rule (optional, repetitive): A constraint that describes specific conditions that are applicable to the data type.

[DOC2]	A data type definition MAY contain one or more content component restrictions to provide additional information on the relationship between the data type and its corresponding UDT. If used, the content component restrictions must contain a structured set of annotations in the following patterns: ♦ RestrictionType (mandatory): Defines the type of format restriction that applies to the content component. ♦ RestrictionValue (mandatory): The actual value of the format restriction that applies to the content component. ♦ ExpressionType (optional): Defines the type of the regular expression of the restriction value.
[DOC3]	A data type definition MAY contain one or more supplementary component restrictions to provide additional information on the relationship between the data type and its corresponding UDT. If used, the supplementary component restrictions must contain a structured set of annotations in the following patterns: ♦ SupplementaryComponentName (mandatory): Identifies the supplementary component on which the restriction applies. ♦ RestrictionValue (mandatory, repetitive): The actual value(s) that is (are) valid for the supplementary component.

The following rule describes the documentation requirements for each BBIE definition.

Documentation Rule	
[DOC4]	Every Basic Business Information Entity (BBIE) definition MUST contain a structured set of annotations in the following patterns: ♦ Unique Identifier (mandatory): The identifier that references a BBIE instance in a unique and unambiguous way. ♦ CategoryCode (mandatory): The category to which the object belongs. In this case, the value will always be BBIE. ♦ Dictionary Entry Name (mandatory): The official name of a BBIE. ♦ Version (mandatory): An indication of the evolution over time of a BBIE instance. ♦ Definition (mandatory): The semantic meaning of a BBIE. ♦ Cardinality (mandatory): Indication whether the BBIE property represents a not-applicable, optional, mandatory, and/or repetitive characteristic of the Aggregate Business Information Entity. ♦ QualifierTerm (optional): Qualifies the property term of the associated core component property in the associated Aggregate Core Component. ♦ UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the BBIE. ♦ ConstraintLanguage (optional, repetitive): A formal description of a way the BBIE is derived from the corresponding stored core component and stored business context. ♦ BusinessTerm (optional, repetitive): A synonym by which the BBIE is commonly known and used in the business. ♦ Example (optional, repetitive): Example of a possible value of a BBIE.

The following rule describes the documentation requirements for each ABIE definition.

Documentation Rule

[DOC5] Every Aggregate Business Information Entity (ABIE) definition MUST contain a structured set of annotations in the following patterns:

- UniqueIdentifier (mandatory): The identifier that references an ABIE instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. In this case, the value will always be ABIE.
- Version (mandatory): An indication of the evolution over time of an ABIE instance.
- DictionaryEntryName (mandatory): The official name of an ABIE.
- Definition (mandatory): The semantic meaning of an ABIE.
- QualifierTerm (mandatory): Qualifies the object class term of the associated Aggregate Core Component.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the ABIE.
- ConstraintLanguage (optional, repetitive): A formal description of a way the ABIE is derived from the corresponding stored core component and stored business context.
- BusinessTerm (optional, repetitive): A synonym by which the ABIE is commonly known and used in the business.
- Example (optional, repetitive): Example of a possible value of an ABIE.

The following rule describes the documentation requirements for each ASBIE definition.

Documentation Rule

[DOC6] Every Association Business Information Entity (ASBIE) definition MUST contain a structured set of annotations in the following patterns:

- UniqueIdentifier (mandatory): The identifier that references an ASBIE instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. In this case, the value will always be ASBIE.
- DictionaryEntryName (mandatory): The official name of an ASBIE.
- Definition (mandatory): The semantic meaning of an ASBIE.
- Version (mandatory): An indication of the evolution over time of an ASBIE.
- Cardinality (mandatory): Indication whether the ASBIE property represents a not-applicable, optional, mandatory, and/or repetitive characteristic of the ABIE.
- QualifierTerm (optional): Qualifies the property term of the associated core component property in the associated aggregate core component.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the ASBIE.

- ConstraintLanguage (optional, repetitive): A formal description of a way the ASBIE is derived from the corresponding stored core component and stored business context.
- BusinessTerm (optional, repetitive): A synonym term under which the ASBIE is commonly known and used in the business.
- Example (optional, repetitive): Example of a possible value of an ASBIE.

The following rule describes the documentation requirements for each Core Component definition.

Documentation Rule

[DOC7] Every Core Component definition MUST contain a structured set of annotations in the following patterns:

- UniqueIdentifier (mandatory): The identifier that references a Core Component instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. In this case the value will always be CCT.
- DictionaryEntryName (mandatory): The official name of a Core Component.
- Definition (mandatory): The semantic meaning of a Core Component.
- ObjectClass: The object class represented by the type.
- PropertyTerm: The property term represented by the type.
- Version (mandatory): An indication of the evolution over time of a Core Component instance.
- Usage Rule (optional, repetitive): A constraint that describes specific conditions that are applicable to the BBIE.
- Business Term (optional, repetitive): A synonym by which the BBIE is commonly known and used in the business.

The following rule describes the documentation requirements for each element declaration.

Documentation Rule

[DOC8] Every element declaration MUST contain an annotation as follows:
<Documentation>

<cdp:[Dictionary Entry Name]></cdp:[Dictionary Entry Name]>

</Documentation>

where Dictionary Entry Name is the complete name (not the tag name) that is the unique official name of the element in the DON library.

The following example is an excerpt from the StrikePlan example in Appendix D, *Schema Examples*.

Example: ABIE, BBIE and ASBIE documentation elements

```
<xsd:element name="Target" type="bie1-0:TargetType"/>
    <xsd:complexType name="TargetType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:ABIEComponent>
                    <cdp:UID>A0002</cdp:UID>
                    <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Target.
                    Details</cdp:DictionaryEntryName>
                    <cdp:Version>1.0</cdp:Version >
                    <cdp:Definition>An unit or location designated for
                    attack.</cdp:Definition>
                    <cdp:ObjectClass>Target</cdp:ObjectClass>
                    <cdp:PropertyTerm>Details</cdp:PropertyTerm>
                    <cdp:RepresentationTerm>Details</cdp:RepresentationTerm>
                </cdp:ABIEComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="TargetID" type="bie1-0:TargetIDType"/>
            <xsd:element ref="bie1-0:Description">
                <xsd:annotation>
                    <xsd:documentation>
                        <cdp:BBIEComponent>
                            <cdp:UID>B0002</cdp:UID>
                            <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                            <cdp:DictionaryEntryName>Target. Description.
                            Text</cdp:DictionaryEntryName>
                            <cdp:Definition>A free-text description of a
                            target.</cdp:Definition>
                            <cdp:Version>1.0</cdp:Version>
                            <cdp:ObjectClass>Target</cdp:ObjectClass>
                            <cdp:PropertyTerm>Description</cdp:PropertyTerm>
                            <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                            <cdp:Cardinality>1.1</cdp:Cardinality>
                        </cdp:BBIEComponent>
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element ref="bie1-0:AssignedOrdnanceREF">
                    <xsd:annotation>
                        <xsd:documentation>
                            <cdp:ASBIEComponent>
```

```
                        <cdp:UID>AS0005</cdp:UID>
                        <cdp:CategoryCode>ASBIE</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Target. Assigned. Ordnance
                        Reference</cdp:DictionaryEntryName>
                        <cdp:Definition>A pointer to an ordnance item that has been
                        assigned to a target.</cdp:Definition>
                        <cdp:Version>1.0</cdp:Version>
                        <cdp:ObjectClass>Target</cdp:ObjectClass>
                        <cdp:AssociationName>Assigned</cdp:AssociationName>
                        <cdp:AssociatedObjectClass>Ordnance
                        Reference</cdp:AssociatedObjectClass>
                        <cdp:Cardinality>1.1</cdp:Cardinality>
                    </cdp:ASBIEComponent>
                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
```

The following rule describes the documentation requirements for each DON construct containing a code.

Documentation Rule

[DOC9] For each DON construct containing a code, the DON documentation MUST identify the zero or more code lists that MUST be minimally supported when the construct is used:

- Prefix (mandatory): The code prefix, for example “cnt” for country code list.
- CodeListQualifier (mandatory): The qualifier for the code list, for example, “ISO 3166-1.”
- CodeListAgency: The maintainer of the code list, for example “6.”
- CodeListVersion: The version of the code list, for example “0.3.”

Section 7
Naming, Definition, and Declaration Rules

Schema language provides many redundant features that allow a developer to represent a logical data model many different ways. Heterogeneous data models can become an interoperability problem without prescribing a comprehensive set of naming, definition, and declaration design rules.

This section establishes rules for XML schema elements, attributes, and type creation. Because the W3C XML specifications are flexible, the DON requires comprehensive rules to achieve a balance between establishing uniform schema design while still providing developers flexibility across the DON.

Adherence to these rules will ensure that semantics are unambiguous, enabling the harmonization team to conduct straightforward comparisons and make recommendations with respect to enterprise reusability across the DON.

7.1 Purpose

The purpose of this section is to summarize the DON XML naming, definition, and declaration rules and approach.

7.2 Naming Rules

7.2.1 General Naming Rules

The English language has many spelling variations for the same word. For example, American English “program” has a corresponding British spelling “programme.” This variation has the potential to cause interoperability problems when exchanging XML components because of the different names used by the same elements. By providing a dictionary standard for spelling, the DON will mitigate this potential interoperability issue.

General Naming Rule	
[GNR1]	DON XML element, attribute, and type names MUST be in the English language, using the Oxford English Dictionary for Writers and Editors (Latest Ed.). Where both American and English spellings of the same word are provided, the American spelling MUST be used.

The CCTS provides a rule set for precisely defining the semantics of a data element in terms of a tripartite naming convention specified by ISO 11179 Part 5 (object class, property term, and representation term). The three parts are combined into a single dictionary entry name by separating each with a dot followed by a space and removing

redundancies between the parts. The names of DON XML elements, complex types, and attributes are derived by rules from the underlying data element's dictionary entry name.

General Naming Rule	
[GNR2]	DON XML element, attribute, and type names MUST conform to CCTS dictionary entry names with all separators and spaces removed.

Example: *Data Element Dictionary Entry Name—"Track. Mission. Code"*

XSD Complex Type–"TrackMissionCodeType"
XSD Global Element–<TrackMissionCode>

To successfully employ the DON rules, information analysis is required to create an information model and define the model elements according to the rules provided by the CCTS. DON XML component names must be constructed with upper and lower case alphabetic characters, a–z and A–Z. Periods, spaces, other separators, or characters are not allowed, and white space is removed.

General Naming Rule	
[GNR3]	DON enterprise XML element, attribute, and type names MUST NOT use abbreviations or other word truncations (e.g. acronyms), except those in the approved list published by the cognizant FNC.

General Naming Rule	
[GNR4]	Abbreviations and acronyms MUST be submitted to an FNC for approval.

Organizations submitting requests for additions to a published list of abbreviations must consider all possible interpretations of the proposal and only submit those that are unambiguous and are likely to have meaning external to the local organization.

General Naming Rule	
[GNR5]	The abbreviations and acronyms list approved by the BSC and FNC MUST be used.

To promote consistency in XML component names, developing organizations must consult lists published by BSC and substitute approved abbreviations in XML component names.

General Naming Rule	
[GNR6]	DON XML element, attribute, and type names MUST be in singular form unless the concept itself is plural (example: goods).

In rare cases, plural names are recommended, but the developer should use discretion to determine which XML component names are plural.

Example: *Plural Usage*

```
<Targets>
    <Target>…</Target>
    <Target>…</Target>
</Targets>
```

In this example, the parent element represents a plural concept. The contents of this element are defined as an unbounded sequence of singular target entries.

General Naming Rules	
[GNR7]	The UpperCamelCase (UCC) convention MUST be used for naming elements and types.
[GNR8]	The lowerCamelCase (LCC) convention MUST be used for naming attributes.

The XML Schema construct xsd:unique is allowed by the DON XML rules to normalize XML instances by expressing one-to-many relationships between model classes/entities. The name of the xsd:unique identity constraint is the object class name of the class/entity being identified uniquely plus the string "Key."

General Naming Rule	
[GNR9]	The xsd:unique identity constraints names MUST be the same as the object class of the ABIE being identified uniquely plus the suffix "Key."

The XML Schema construct xsd:keyref is used in conjunction with xsd:unique to identify relationships between model elements. The name of xsd:keyref constraints is constructed by concatenating the referencing object class to the referenced object class and appending the string "REFKey."

General Naming Rule	
[GNR10]	The xsd:keyref identity constraint names MUST be consist of the name of the referencing object class plus the name of the referenced object class plus the suffix "REFKey."

Example: *A 'StrikeType' xsd:complexType is defined to represent a 'Strike' class/entity. In the model, the Strike class has relationships between some of its child classes that would benefit from instance normalization:*

```
<xsd:complexType name="StrikeType">
        <xsd:sequence>
            <xsd:element ref="TargetList"/>
            <xsd:element ref="AvailableOrdnanceList"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:element name="Strike" type="StrikeType">
<xsd:unique name="OrdnanceKey">
        <xsd:selector xpath="AvailableOrdnanceList/Ordnance"/>
        <xsd:field xpath="OrdnanceID"/>
</xsd:unique>
<xsd:keyref name="TargetOrdnanceREFKey" refer="OrdnanceKey">
        <xsd:selector xpath="TargetList/Target/AssignedOrdnance"/>
        <xsd:field xpath="@ordnanceIDREF"/>
</xsd:keyref>
</xsd:element>
```

The <TargetList> and <AvailableOrdnanceList> elements contain sequences of <Target> and <Ordnance> elements. The schema designer desires to enable referencing a single <Ordnance> element. Since the <Ordnance> element contains a unique reference, the xsd:unique identity constraint is named 'OrdnanceKey' (object class 'Ordnance').

The <Target> element is referencing unique <Ordnance> elements, therefore the xsd:keyref constraint is named 'TargetOrdnanceREFKey' (from 'Target' object class to 'Ordnance' object class).

7.2.2 Element Naming Rules

ABIEs defined in information analysis are represented in the XML schema language as both XML schema complex data types (xsd:complexType) and global elements. The xsd:complexType is normally defined first, then the global element is declared to be of the xsd:complexType. The name of the global element is derived directly from the name of the xsd:complexType.

Element Naming Rule	
[ELN1]	A DON CCTS:ABIE element name MUST be the same as the corresponding xsd:complexType to which it is bound, with the word "Type" removed.

Example: *ABIE Dictionary Entry Name: 'Address. Details'*

xsd:complexType Name: "AddressType"
Global Element Name: <Address>

BBIEs identified in information analysis must be expressed as global elements (see Rule ELD1), although local elements (ID, Code, Measure) are allowable in some cases. Like

global elements, BBIE xsd:complexTypes are normally defined, then the elements are declared to be of that type. BBIE element names are constructed by removing the word "Type" from the xsd:complexType name. If the resultant name is overly ambiguous, the object class name can be added to provide semantic clarity.

Element Naming Rule	
[ELN2]	A DON global element based on a CCTS:BBIE Property element MUST be the same name of its corresponding xsd:complexType with the word "Type" removed, unless the object class can be used for semantic clarity.

For local elements, the object class will usually be removed because the parent type provides context. Because global elements may be employed independent of a parent xsd:complexType, the object class is more likely to be required for clarity.

Example: *BBIE Dictionary Entry Name: 'Ordnance. Identifier'*

xsd:complexType Name: 'OrdnanceIDType'
Element Name: <OrdnanceID>
The object class name is required to provide better clarity of 'what' <ID> is being declared.
Example 2:
BBIE Dictionary Entry Name: 'Standard. Country. Code'
xsd:complexType Name: 'StandardCountryCodeType'
Element Name: <CountryCode>
In this case, the object class name is dropped as it is not required to convey the meaning of a country code.

As a final example, it may be necessary to add an object class to a previously defined BBIE global element name to differentiate it from another during the harmonization process.

Example: Creating a Global Element <Nomenclature> to Represent the BBIE 'Ordnance. Nomenclatures. Text'

<Ordnance>
<OrdnanceID>OrdanceID001</OrdnanceID>
<Nomenclature>UBS 001A</Nomenclature>
</Ordnance>

During harmonization, it is discovered that the term 'nomenclature' is discovered that the C4I acquisition community uses the term nomenclature to refer to equipment. Because of harmonization, the element <Nomenclature> is deprecated and two new global elements are created - <OrdnanceNomenclature> for 'Ordnance. Nomenclature. Text' and <EquipmentNomenclature> for 'Equipment. Nomenclature. Text'.

<Ordnance>
<OrdnanceID>OrdanceID001</OrdnanceID>
<OrdnanceNomenclature>UBS 001A</Nomenclature>
</Ordnance>
<Equipment>
<EquipmentID>EquipmentID001</EquipmentID >
<EquipmentNomenclature>AN/UQS-123</EquipmentNomenclature>
</Equipment>

An ASBIE name must be based on the ASBIE dictionary entry name and then may add an optional suffix, depending on the type of association being expressed. If the association cardinality is greater than one, the element name may be made plural or the word "List" can be appended.

Element Naming Rule

[ELN3] A DON CCTS:ASBIE name MUST be based on the CCTS:ASBIE dictionary entry and MAY contain an optional suffix to provide clear cardinality.

The ASBIE dictionary entry name is the name of the model class/entity from which the association originates, the object class name of the model class being associated to, and a property term.

Example: *ASBIE Dictionary Entry Name: 'Mission. Assigned. Ordnance'*

Possible ASBIE Global Element Names:

<AssignedOrdnance> Typically used when there is only one 'Assigned Ordnance' (e.g. one to one relationship)

<AssignedOrdnances> - Typically used when there is more than one 'Assigned Ordnance'

(e.g. one to one relationship)

<AssignedOrdnanceList> - Allowable Option

According to the DON schema modularity, namespace, and versioning strategy, Root-Level Schema modules are created to define a document model for the publication of content (document-oriented) or to define a set of messages that are exchanged within a business process (data-oriented). Thus, the root global element name can represent either

the name (or title) of the document being published or the name of the business or mission process.

Element Naming Rule	
[ELN4]	Each root element in a DON Root Schema module document MUST be named according to the portion of the business process initiated or the content item published.

Example: *Naming the Root Element*

Document Title: DON XML Development Guide
Root Element Name: <DONXMLDevelopmentGuide>
– –
Example:
Process Name: Stock Check
Process Parts: Stock Check Request & Stock Check Response
Root Element Names: <StockCheckRequest>, <StockCheckResponse>

7.2.3 Attribute Naming Rules

Rule ATD1 strongly discourages the use of user-defined attributes except when employing the CCTS:SupplementaryComponent attributes declared in the DON Qualified Data Types module. Because these attributes are already defined and available for DON developers to reuse, the NDR does not provide detailed rules for attribute naming.

7.2.4 Complex Type Naming Rules

ABIEs defined in information analysis are represented in the XML schema language as both XML schema complex data types (xsd:complexType) and global elements. The xsd:complexType is normally defined first, then the global element is declared to be of the xsd:complexType. The name of the global element is derived directly from the name of the xsd:complexType. The name of the xsd:complexType is based on the CCTS:DictionaryEntryName.

Complex Type Naming Rule	
[CTN1]	A DON xsd:complexType name based on a ccts:ABIE MUST be the ccts:DictionaryEntryName object class, property term, any qualifiers and the representation term with the separators removed and the “Details” suffix replaced with “Type.”

DON xsd:complexType names for cctsBasicBusinessInformationEntities will be derived from their dictionary entry name by removing the object class (allowed under truncation rules), removing separators to follow general naming rules, and appending the suffix “Type.”

Complex Type Naming Rule	
[CTN2]	A DON xsd:complexType name based on a ccts:BBIE Property MUST, at a minimum, include the ccts:DictionaryEntryName property term, any qualifiers and the representation term, with the separators removed and with the "Type" suffix appended after the representation term.

A DON developer has the flexibility to include the object class into the xsd:complexType name, if necessary to better define the semantics or context of the BBIE.

Complex Type Naming Rule	
[CTN3]	A DON xsd:complexType name based on a ccts:QualifiedDataType (QDT) MUST, at a minimum, include the dictionary entry name of the ccts:UnualifiedDataType (UDT) it derives from, with the separators removed.

The following is an example of the above rule. TargetIDType must, at a minimum, include the dictionary entry, IdentifierType, of the unqualified data type from which it derives. In this case, IdentifierType is abbreviated as IDType, and the object class, Target, is added to provide better context that can be reused across the DON.

Example: *Data Type Inheritance*

```
<!—Qualified DT: Target_ Identifier. Type—>
<xsd:complexType name="TargetIDType">…
<!—Unqualified DT: Identifier. Type—>
<xsd:complexType name="IDType">
```

Complex Type Naming Rule	
[CTN4]	A DON ccts:DataType name MUST be the ccts:DictionaryEntryName with the separators removed.

7.3 Definitions and Declarations Rules

7.3.1 Complex Type Definition

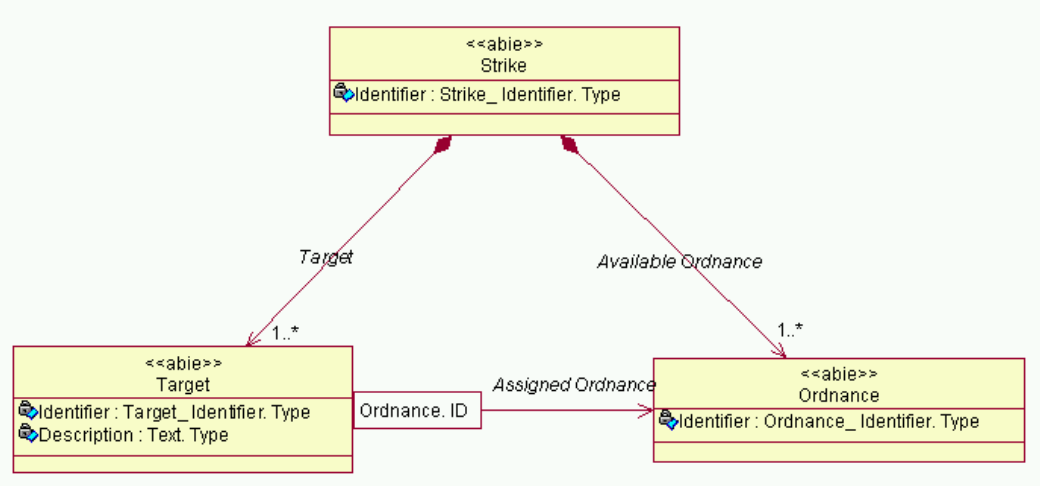
The fundamental XML schema component used as the basis for reuse and interoperability is the *named* complex type (xsd:complexType).

7.3.1.1 ABIE Definition

Complex Type Definition Rule	
[CTD1]	Every ccts:ABIE xsd:complexType MUST use xsd:sequence or xsd:choice with global element references and/or local element declarations to reflect each property of its class.

In the example, the Strike class (ABIE "Strike. Details") has one BBIE property and two ASBIE properties. The "StrikeType" xsd:complexType expresses this by creating an xsd:sequence with references to three global elements.

Example: *XML Schema Representation of 'Strike. Details'*



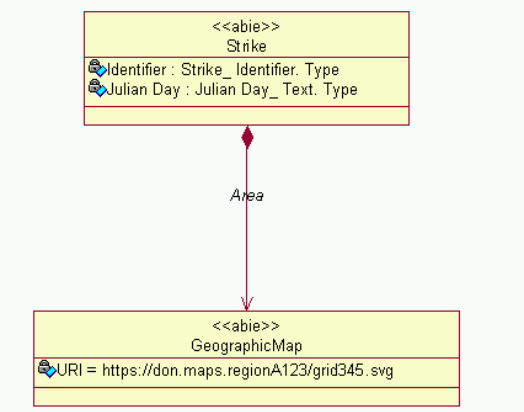

```
<!—======= Aggregate BIEs ======—>
<!—ABIE: Strike. Details—>
<xsd:complexType name="StrikeType">
        <xsd:sequence>
            <xsd:element ref="StrikeID"/>
            <xsd:element ref="Target" maxOccurs="unbounded"/>
            <xsd:element ref="AvailableOrdnance" maxOccurs="unbounded"/>
        </xsd:sequence>
</xsd:complexType>
```

Their may be occasions where the content model of an ABIE xsd:complexType is defined by another XML schema vocabulary. Provided the XML language is BSC approved, the XML schema <xsd:any> feature may be employed to specify the namespace restriction on the content model.

Complex Type Definition Rule	
[CTD2]	The xsd:any MAY be used in the content model of an DON ABIE type when the ABIE represents an object defined by an external XML business standard approved by the DON BSC.

In the example, the ccts:ASBIE "Strike. Area. Geographic Map" is based upon the ABIE "Geographic Map. Details." The contents of "GeographicMapType" created according to rule [CTD1] are constrained by the Scalable Vector Graphics namespace. The <xsd:any namespace="…"/> element expresses this.

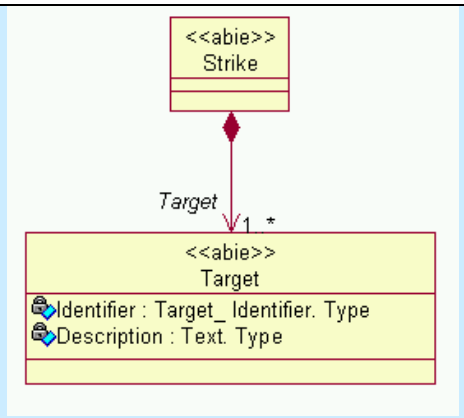
Example: *ASBIE–Strike. Area. Geographic Map —> ABIE–Geographic Map. Details*




```
<!—=====Aggregate BIEs=====—>
<!—ABIE Geographic Map. Details—>
<xsd:complexType name="GeographicMapType">
        <xsd:sequence>
            <xsd:any namespace="http://www.w3.org/2000/svg"/>
        </xsd:sequence>
</xsd:complexType>
```

This rule established the mapping between DON information model ABIEs (classes), data types, and xsd:complexTypes. The rule implies that there is a one-to-many correspondence between BIEs, data types, and named xsd:complexTypes.

Complex Type Definition Rule	
[CTD3]	A named xsd:complexType MUST be defined for every ABIE and BBIE Property identified in a DON information model.



XML Schema Complex Types:

ABIEs:
<!—ABIE: Strike. Details—>
<xsd:complexType name="StrikeType">…

<!—ABIE: Target. Details—>
<xsd:complexType name="TargetType">…
<!—Qualified DT: Target_ Identifier. Type—>

Qualified Data Type:
<xsd:complexType name="TargetIDType">…
<!—Unqualified DT: Identifier. Type—>

Unqualifed Data Type:
<xsd:complexType name="IdentifierType">…
<!—Unqualified DT: xsd:normalizedString type—>

**Note: Data Types Definition rules will further explain the creation of Data Types.

In this example, a named xsd:complexType is created for each ABIE (object class) in the model, and for the data types of the BBIEs. Note that the BBIEs are based on the representation terms "Identifier" and "Text."

7.3.1.2 BBIE Type Definition

All BBIEs are defined by complexType.

Complex Type Definition Rule
[CTD4] Every BBIE property MUST define a named xsd:complexType.

Example: *BBIE That Shares the Same Property Term and Representation Term*

BBIE–Ordnance. Description. Text
BBIE–Target. Description. Text
<xsd:complexType name="DescriptionTextType"

In this example, the pair "Description. Text" is a unique pair used between two different BBIEs. According to rule [CTD2], only one xsd:complexType is defined.

In some cases, it may not be possible to identify a unique BBIE property and representation term pair independent of the object class. This is most often the case when the property and the representation term pair are the same.

Example: *BBIE–Target. Identifier*

```
    <!—===== Aggregate BIEs =====—>
    <!—ABIE Target. Details—>
        <xsd:complexType name="TargetType">…
…
    <!—BBIE: Target. Identifier—>
        <xsd:complexType name="TargetIDType">
```

In this case, the property and representation term pair (Identifier. Identifier) are non-unique and unambiguous. In this case, an xsd:complexType "TargetIDType" is created, and there is a one-to-one mapping between the BBIEs and the xsd:complexType.

Complex Type Definition Rule	
[CTD5]	Every ccts:BBIE Property complexType definition must be based on a complex type representing either a ccts:QualifiedDataType or ccts:UnqualifedDataType.

As illustrated in the examples, the xsd:complexTypes defined to represent BBIEs all have an xsd:type attribute that binds it to an underlying qualified or unqualified data type xsd:complexType.

The allowable content model of a ccts:BBIE xsd:complexType depends on the UDT basis of the BBIE.

Complex Type Definition Rule	
[CTD6]	Every ccts:BBIE Property complex type content model MUST use xsd:simpleContent.

For ccts:BBIEs not based on the "Code. Type" (code) and "Identifier. Type" (ID) UDTs, the content model must be defined using the xsd:simpleContent element. For code and ID BBIEs, this rule does not apply, allowing the content models of these xsd:complexTypes to be defined using the xsd:complexContent.

Code and ID BBIEs content models are allowed to be complex in order to support referencing a choice of global elements representing standardized code list and ID schemes in accordance with rule.

Complex Type Definition Rule	
[CTD7]	Every ccts:BBIE Property xsd:simpleContent element MUST contain either the xsd:extension or xsd:restriction element.

The ccts:BBIE xsd:complexTypes with simple content must either extend or restrict a ccts:DataType xsd:complexType. Extensions and restrictions are used to change parameters of ccts:Supplementary Component Attributes.

Example: *BBIE—Target. Description. Text*

```
<!—===== Basic BIE Properties =====—>
<xsd:complexType name="DescriptionType">
        <xsd:simpleContent>
            <xsd:extension base="udt:TextType"/>
        </xsd:simpleContent>
</xsd:complexType>
```

The property and representation term of this BBIE, "Description. Text," forms a reusable BBIE property that is defined as an xsd:ComplexType. The complex type uses xsd:simpleContent and xsd:extension to define the xsd:complexType's base as "udt:TextType." The "udt" namespace prefix indicates that "Text. Type" is defined in the Unqualified Data Types Schema module, it is a primary representation term for which an unqualified ccts:DataType has been created.

Another example of a BBIE that is represented as complex type where the property and representation term are not unique is shown below. The xsd:complexType "OrdnanceIDType" is defined as a simple content extension based upon a qualified ccts:DataType of the same name, "qdt:OrdnanceIDType." The "qdt" namespace indicated that the type is defined in the DON Enterprise Qualified Data Types namespace; the prefix differentiates between the two xsd:complexTypes of the same name.

Example: *BBIE—Ordnance. Identifier*

```
<!—===== Aggregate BIEs =====—>
<!—ABIE Ordnance. Details—>
<xsd:complexType name="OrdnanceType">…
…
<!—BBIE: Ordnance. Identifier—>
<xsd:complexType name="OrdnanceIDType">
        <xsd:simpleContent>
            <xsd:extension base="qdt:OrdnanceIDType"/>
        /xsd:simpleContent>
</xsd:complexType>
```

Complex Type Definition Rule	
[CTD8]	Code and ID ccts:BBIE Property complex types MAY use the xsd:choice element to reference global elements defined in standardized ID Scheme or Code List Schema modules.

BBIE xsd:complexTypes that are not based on code or ID must have simple content, whereas those types that are based on code or ID may have complex content. This rule supports the developing OASIS specification for expressing code lists and separate schema modules. The DON anticipates that this same specification will be adapted to ID scheme as well. The content models of code- and ID-based BBIE types can have content models that are a choice of global elements representing code lists or ID schemes defined in their own schema modules.

Example: *BBIE–Target. Country. Code*

```
<!—===== Basic BIE Properties =====—>
<xsd:complexType name="CountryCodeType">
    <xsd:choice>
        <xsd:element ref="iso3166-1:ISO31661CountryCode"/>
    </xsd:choice>
</xsd:complexType>
```

The example BBIE, CountryCode, is a code and is not required to have simple content. In this example, only one global element is referenced in the choice, "iso3166-1:ISO31661CountryCode." This global element is defined in an IS0 3166 Country Code List Schema module as indicated by the namespace prefix. Choice was used so that the xsd:complexType can later be extended to include in the choice other Code List Schema module global elements if the ISO3166-1 code list is not sufficient.

7.3.1.3 DataType Definition

The information analysis section of this document describes the concept of data types as specified in the CCTS (ccts:DataTypes). This section of the NDR provides rules for creating XML schema Complex Types (xsd:complexTypes) to represent the Syntax Neutral Model ccts:DataTypes identified in information analysis.

Complex Type Definition Rule	
[CTD9]	EVERY primary and secondary representation term MUST have an xsd:complexType defined in the DON Enterprise Unqualified ccts:DataTypes Schema module (UDT)

The CCTS identifies primary and secondary representation terms for each CCT. This rule establishes a core set of unqualified ccts:DataTypes mapping to these terms. These xsd:complexTypes serve as the basis for BBIE xsd:complexTypes and qualified ccts:DataType xsd:complexTypes. The DON XML BSC is responsible for publishing and maintaining the approved Unqualified ccts:DataTypes Schema module.

The xsd:complexTypes representing ccts:DataTypes are named according to the data type's dictionary entry name.

Example: *Data Type - Hostile Target_ Identifier. Type*

<xsd:complexType name="**HostileTargetIDType**">…

Complex Type Definition Rule

[CTD10] Every ccts:DataType complex type MUST use xsd:simpleContent.

Unlike BBIE xsd:complexTypes, xsd:complexTypes representing ccts:DataTypes must have only simple content.

Complex Type Definition Rule

[CTD11] Every ccts:DataType SHOULD use xsd:restriction.

Unlike BBIE xsd:complexTypes, xsd:complexTypes representing ccts:DataTypes must derive their content models based on restriction only.

Complex Type Definition Rule

[CTD12] Every Qualified ccts:DataType MUST use the xsd:base attribute to define another ccts:DataType.

The CCTS requires qualified cct:DataTypes to define domain restrictions. These domain restrictions may be on existing qualified ccts:DataTypes or on otherwise unrestricted unqualified ccts:DataType based on the primary and secondary representation term.

Example: *xsd:base attribute*

```
<xsd:extension base="qdt:OrdnanceIDType"/>
```

The base attribute references the OrdnanceIDType defined in the "qdt" (qualified data types) namespace as follows:

Example: *Qualified Data Type - OrdnanceIDType*

```
<xsd:schema targetNamespace="urn:us:gov:dod:don:enterprise:qdt:1:0:beta" …
xmlns:udt="urn:us:gov:dod:don:enterprise:udt:1:0" …>
    <xsd:complexType name="OrdnanceIDType">
        <xsd:simpleContent>
            <xsd:restriction base="udt:IdentifierType">
                <xsd:pattern value="[A-Z]\d{5}"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
```

In this example, the xsd:complexType "OrdnanceIDType" (representing the qualified DataType Ordnance_ Identifier. Type) is based on the xsd:complexType "udt:IdentifierType" defined in version 1.0 of the Unqualified Data Type Schema module. The xsd:complexType defines a domain restriction expressed by a regular

expression pattern "[A-Z]\d{5}" meaning, "One capitalized alphabetic followed by 5 digits."

Complex Type Definition Rule	
[CTD13]	Each ccts:SupplementaryComponent xsd:attribute user-defined xsd:simpleType MUST only be used when the ccts:SupplementaryComponent value is based on a standardized code list for which a BSC approved has been created.

The CCTS definition requires that each element of data (BBIE) be based on a Core Component Type. The DON has included the fundamental CCTs in the DON Enterprise UDT module. The CCT Schema module is a normalized Schema for referencing the underlying CCTS data types. The CCT module is for reference only and will not be imported by the DON Enterprise modularity and namespace architecture.

Example: *Core Component Type–Measure. Type*

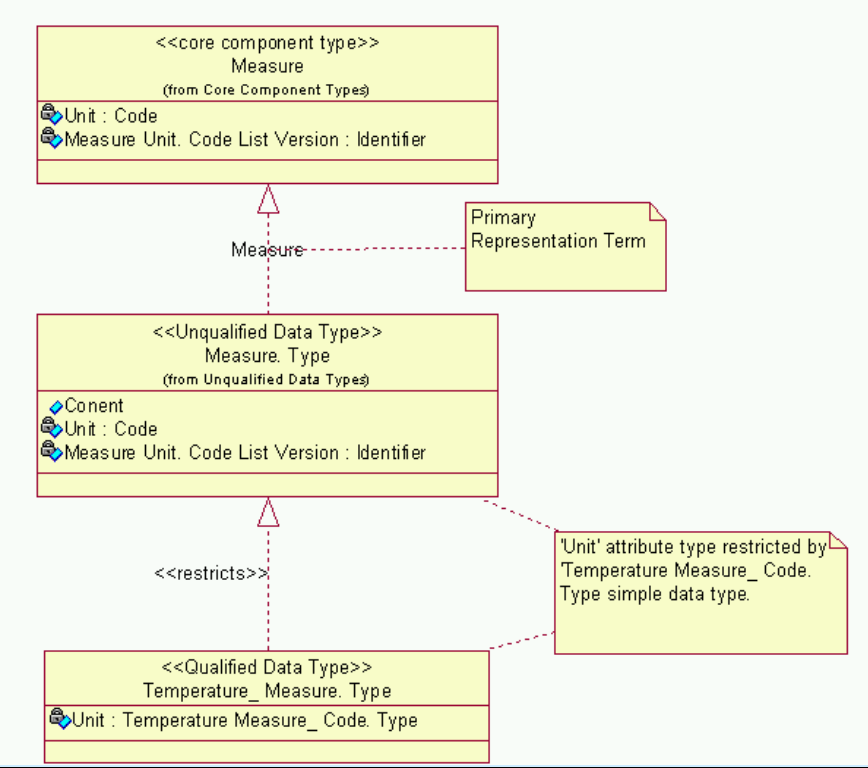
Content Component–
    Measure. Content
Supplemental Components–
    Measure Unit. Code
    Measure Unit. Code List Version. Identifier

The CCT Schema module defines a set of xsd:attributes representing these supplemental components. The attributes are defined in the content model of xsd:complexTypes representing CCTs. The xsd:complexTypes in the Unqualified ccts:DataTypes Schema module are based[1] on the CCT xsd:complexTypes, thus they include these attributes. Because qualified ccts:DataType or ccts:BBIE xsd:complexTypes are based on unqualified ccts:DataType xsd:complexTypes, they inherit these attributes even though the CCT Schema module will not be imported.

[1] The CCT Schema module will *not* be imported into the DON Enterprise modularity and namespace architecture.

The following diagram illustrates:

Example: *Data Type Inheritance*



BBIEs and datatypes may extend or restrict their underlying complex type. In addition, it is possible to assign user-defined xsd:simpleTypes to these inherited attributes. This is allowed only under the condition that the user-defined xsd:simpleType resides in the BSC-approved Code List or Identifier Schema module. This rule prevents developers from restricting the value of supplementary component attributes with xsd:simpleTypes that are not standardized. If developers want to employ these attributes and restrict their values via xsd:simpleTypes, they should work through their cognizant FNC to identify or develop an international, federal, DoD, or DON enterprise Code List or Identifier Scheme Schema module and obtain BSC approval.

The following schema excerpt is an example of an xsd:complexType defined to represent the qualified ccts:DataType "Temperature_ Measure. Type":

Example: *Qualified Data Type–Temperature_ Measure. Type*

```
<!—Qualified DataType Temperature_ Measure. Type—>
<xsd:complexType name="TemperatureMeasureType">
        <xsd:simpleContent>
        <xsd:restriction base="udt:MeasureType">
            <xsd:fractionDigits value="2"/>
            <xsd:totalDigits value="5"/>
            <xsd:attribute name="unitCode"
type="tmuc:TemperatureMeasureUnitCode"/>
        </xsd:restriction>
    </xsd:simpleContent>
</xsd:complexType>
```

This xsd:complexType is a restriction of the xsd:complexType "udt:MeasureType" (representing the unqualified ccts:DataType "Measure. Type" defined in the Unqualified Data Types Version 1.0 namespace).

It restricts the base type's domain and assigns a user-defined xsd:simpleType "tmuc:TemperatureMeasureUnitCode" to the "unitCode" attribute that this type inherited. Rule [CTD13] requires that this user-defined simple type be created in a BSC-approved Code List Schema module; consequently the "TemperatureMeasureUnitCode" attribute is prefixed with a "tmuc" qualifier, which indicates that the type is defined in the Temperature Measure Unit Code List Schema Module Version 1.0.

Complex Type Definition Rule	
[CTD14]	Every Qualified ccts:DataType (QDT) xsd:complexType content model xsd:restriction element MUST use the xsd:base attribute to define the basis as a UDT xsd:complexType.

As illustrated in the above examples, xsd:complexTypes representing qualified ccts:DataTypes must employ an xsd:restriction element with a base attribute referencing an xsd:complexType defined in the Unqualifed DataTypes (UDT) Schema module.

The ccts:SupplementaryComponent xsd:attribute usage must not be "prohibited."

Complex Type Definition Rule	
[CTD15]	Each ccts:SupplementaryComponent xsd:attribute "use" MUST define the occurrence as either "required" or "optional."

7.3.2 Element Declarations

7.3.2.1 General Element Declarations

Elements declared for BBIEs that are based on the UDTs "Identifier. Type," "Measure. Type," or "Code. Type" may not be globally reusable. In other words, identification, measurement, or code list schemes may be highly specialized and not managed at an enterprise level. In these cases, it is more appropriate to define these BBIE elements as local child elements of xsd:complexTypes/global elements representing the parent ABIE. Although developing organizations should make a recommendation as to the scope of a code, identifier, or measure element, the cognizant FNC must make the final determination.

Element Declaration Rule	
[ELD1]	All element declarations MUST be global with the exception of `Identifiers, Measures, and Codes`, which MAY be declared as local elements if, and only if, approved by the FNC and BSC.

Example: *Local Element Declaration—TargetID*

```
<xsd:complexType name="TargetType">
    <xsd:sequence>
        <xsd:element name="TargetID" type="bie:TargetIDType"/>
        <xsd:element ref="bie:AssignedOrdnanceREF"/>
    </xsd:sequence>
</xsd:complexType>
```

In this example, the element <TargetID> is created as a local element due to the FNCs determination that the scope of the Target identification scheme is specific only to the developing organization.

7.3.2.2 Elements Bound to Complex Types

Element Declaration Rule	
[ELD2]	For every xsd:complexType representing a CCTS:ABIE, a global element MUST be declared.
[ELD3]	For every xsd:complexType representing a CCTS:BBIE Property, a global element MUST be declared.
[ELD4]	For each CCTS:ASBIE, a global element MUST be declared.
[ELD5]	For CCTS:BBIEs that are based on ID, code, and measure, a local element MAY be declared in the xsd:complexType of the parent ABIE.

Example: Element Declaration

XML Schema xsd:complexType and global element definition for 'Target. Details' ABIE:

```
<xsd:element name="Target" type="TargetType"/>
<xsd:complexType name="TargetType">
    <xsd:sequence>
        <xsd:element name="TargetID" type="TargetIDType"/>
        <xsd:element ref="TargetDescription" type="TargetDescriptionType"/>
        <xsd:element ref="AssignedOrdnance"/>
    </xsd:sequence>
</xsd:complexType>
```

This example shows a **<Target>** global element bound to the **'TargetType'** xsd:complexType declared for a 'Target. Details' ABIE [rule ELD3].
The ABIE has two component BBIEs, a 'Target. Identifier' and a 'Target. Description. Text', and one ASBIE, 'Target. Assigned. Ordnance'.
The <TargetID> BBIE element is defined locally, the **<TargetDescription>** BBIE element is defined globally.
The ASBIE global element **<AssignedOrdnance>** is declared directly, rather than being bound to a type.

7.3.2.3 Empty Elements

Empty elements may cause application or XML processing difficulty and must be avoided except for the following cases:

- Reference elements (either local or global), which contain Object Identification Reference Attributes, serve as pointers to elements elsewhere in an instance and therefore are themselves empty.
- Xlink elements, which are declared in accordance with the Xlink specification, may be empty.

Element Declaration Rule	
[ELD6]	Empty elements SHALL NOT be declared except for reference elements and Xlink elements, which MUST be approved by the cognizant FNC and BSC.

Example: *The XML Instance for the < Target>*

```
<Target>
    <TargetID>AX001</TargetID>
    <TargetDescription></TargetDescription>
    <AssignedOrdnanceREF ordnanceIDREF="OrdanceID001"/>
</Target>
```

Example: *Xlink Empty Element*

<AssignedOrdnanceLink xlink:href="Ordnance.xml#*[@ordnanceID='001A']"/>
In this example, the ASBIE Global Xlink Element is empty, containing an Xlink attribute referencing an element in an external XML document.

7.3.3 Attribute Declarations

XML attributes provide an alternative means to express information in a document instance. The choice between elements and attributes is a design decision with no universally accepted best practice. The DON has chosen to adopt the principle that attributes should be used only to express metadata about business or mission execution data carried as elements. This principle minimizes attribute usage and has several desirable advantages:

- It harmonizes the DON approach with that of major VCS XML business standard efforts.
- Because XML processors handle attribute values somewhat differently than element values, it minimizes the possibility of interoperability errors by ensuring that all business or mission execution data are handled consistently as elements.
- It ensures that DON enterprise XML has a consistent element/attribute structure.

Attribute Declaration Rule

[ATD1] User-defined attributes SHOULD NOT be used.

All ccts:BBIE elements must be based on an underlying DON Enterprise Unqualified Datatype (UDT). As shown in the next example, the BBIE '*Strike. Surface_ Temperature. Measure'* is based on the UDT *'MeasureType.'*

Example: A BBIE Global Element Based on the Measure Unqualified Data Type
XML Instance:

```
<Strike>
    <SurfaceTemperatureMeasure unitCode="DON1">67</SurfaceTemperatureMeasure>
<Strike>
————————————————————————-
```

Dictionary Entry Name
BBIE–'Strike. Surface_ Temperature. Measure'

```
————————————————————————-
```

Qualified DataType–'Surface_Temperature. Measure. Type'

```
<xsd:element name="SurfaceTemperatureMeasure" type="qdt1-
0:SurfaceTemperatureMeasureType"/>
<xsd:complexType name="SurfaceTemperatureMeasureType">
    <xsd:simpleContent>
        <xsd:extension base="qdt1-0:TemperatureMeasureType"/>
    </xsd:simpleContent>
</xsd:complexType>
```

```
<!—This element and complexType also created in the QDT version 1.0 namespace—>
<xsd:complexType name="TemperatureMeasureType">
        <xsd:simpleContent>
            <xsd:restriction base="udt1-0:MeasureType">
                <xsd:fractionDigits value="2"/>
                <xsd:totalDigits value="5"/>
            </xsd:restriction>
        </xsd:simpleContent>
</xsd:complexType>
———————————————————-
Unqualified DataType–'Measure. Type'
<!—This element and complexType created in the UDT version 1.0 namespace—>
<!—===== UDT: MeasureType =====—>
    <xsd:element name="Measure" type="MeasureType"/>
    <xsd:complexType name="MeasureType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0013</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Measure.
Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A numeric value determined by measuring an object
along with the specified unit of measure.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Measure</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:decimal">
                <xsd:attribute name="unitCode" type="xsd:token" use="optional"/>
                <xsd:attribute name="unitCodeListVersionID" type="xsd:token"
use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
```

This example illustrates how the "unitCode" attribute in an XML instances is initially defined in the "MeasureType" xsd:complexType in the Core Component Type XML Schema module.

Attribute Declaration Rule	
[ATD2]	If a DON `xsd:SchemaExpression` contains one or more common attributes that apply to all DON elements contained or included or imported therein, the common attributes MUST be declared as part of a global attribute group.

This rule allows for grouping of common attributes to simplify reuse.

Attribute Declaration Rule	
[ATD3]	An objectIDREF xsd:attribute MUST be declared globally in the DON Enterprise BIE Reusable Schema module.

The section on modeling discusses the concept of reference associations. Rules [CTD1], [CTD19], and [ELD6] describe the requirement to create xsd:complexTypes and global elements for each reference association identified in business information analysis. Reference associations reference other points in an XML instance via an object identification reference (objectIDREF) attribute declared in the content model of the reference association xsd:complexType. Because many reference associations potentially exist between model elements, the objectIDREF attribute is declared globally in the DON Enterprise BIE Reusable Schema module.

Attribute Declaration Rule	
[ATD4]	The objectIDRef xsd:attribute value MUST be equal to the value of an ID ccts:BBIE element.

As its name implies, the objectIDREF attribute must uniquely identify another model object within an instance. To ensure that this is done consistently, the value of this attribute MUST be equal to the value of a ccts:BBIE global element based on the identifier UDT.

Example: *Sample - Object ID Reference Attribute*

```
<Strike>
    <Target>
        <AssignedOrdnanceREF objectIDREF="A12345"/>
    </Target>
    <AvailableOrdnance>
        <OrdnanceID>A12345</bie1-0:OrdnanceID>
    </AvailableOrdnance>
</Strike>
```

In this example, the <AssignedOrdnanceREF> element has an objectIDREF attribute that references the <OrdnanceID> element. This construct allows the XML instance to be normalized by creating a pointer in the <Target> element to an <AvailableOrdnance> item.

Section 8
Code List Rules

The DON has determined that the best approach for code lists is to handle them as schema modules. Because most code lists are maintained by external agencies, the DON has determined that if code list owners all use the same normative form schema module, all users of those code lists could avoid a significant level of code list maintenance. A key guiding principle is that "the DON library SHOULD identify and use external standardized Code List Schema modules rather than develop its own DON-native Code List Schema modules."

Externally developed and owned code lists will be imported into the DON Control Schema module. The maintenance and development of these external code lists will remain the responsibility of their owners.

The code list strategy uses several different kinds of schema modules:

- External Code List Schema module
- DON Code List Schema module.

These modules are imported into the Control Schema module, and each resides in its own namespace.

8.1 Purpose

The purpose of this section is to summarize the DON XML code list rules and approach.

8.2 Code List Rules

To make this mechanism operational, DON has defined a number of rules:

Code List Rule	
[CDL1]	All codes used in DON MUST be part of a DON-maintained or externally maintained Code List Schema module.

Because the majority of code lists are owned and maintained by external agencies, DON will make maximum use of such external Code List Schema modules where they exist.

In some cases, the DON library may extend an existing code list to meet specific business requirements. In others cases the DON library may have to create and maintain a code list when a suitable code list does not exist in the public domain. Both of these types of code lists would be considered DON-internal code lists.

Code List Rule	
[CDL2]	The DON library MAY design and use an internal code list if an existing external code list needs to be extended or if no suitable external code list exists.

DON-internal code lists will be designed with maximum reuse in mind to facilitate maximum use by others.

If a DON code list is created, the lists should be scoped for reuse (designed for reuse and sharing, using named types and schema modules) rather than locally scoped (not designed for others to use and therefore hidden from their use).

To guarantee consistency within all Code List Schema modules, all DON-internal code lists and externally used code lists will use the DON Code List Schema module. This schema module will contain an enumeration of code list values.

Code List Rule	
[CDL3]	All DON-maintained or used code lists MUST be enumerated using the DON Code List Schema module.

To guarantee consistent naming of Code List Schema modules, the name of each DON Code List Schema module will adhere to a prescribed form.

Code List Rule	
[CDL4]	The name of each DON Code List Schema module MUST be: <Owning Organization>-Codelist-<Code List Name>- <Version>.

Example: *Two possible Code Lists.*

DON-Enterprise-CodeList-TemperatureMeasureUnitCode-1-0.xsd
ISO3166-1-CodeList-CountryIdentificationCode-1-0.xsd

Each code list used in the DON schema MUST be imported individually.

Code List Rule	
[CDL5]	An xsd:Import element MUST be declared for every code list required in a DON schema.

The DON library allows partial implementations of code lists that may required by developers.

Code List Rule	
[CDL6]	Users of the DON library MAY identify any subset they wish from an identified code list for their own trading community conformance requirements.

The DON Namespace and Modularity rules detail the specification and syntax of URNs.

The following rule describes the semantic requirements for the namespace of each DON Code List Schema module. The URN consists of some fixed tokens, the name of the code list, and the supplementary components of the code list datatype.

Code List Rule	
[CDL7]	The namespace name of each DON Code List Schema module MUST conform to the following pattern: urn:dod:don:enterprise:codeList:<Code List.Identification.Identifier>:<Code List.Name.Text>:<Code List.Version.Identifier>:<Code List.Agency.Identifier>:<Code List. AgencyName.Text>.

The first three levels are fixed by URN, as defined in the RFC specification:

- urn—the leading token of URNs
- dod—the registered namespace ID "dod"
- don—the registered namespace ID "DON."

The values of the following tokens are determined by the code list being used:

- codeList—DON Code List Schema module
- Code List. Identification. Identifier—list of the respective corresponding codes
- ListID—code list that is unique only within the agency that manages it
- Code List. Name. Text—list of codes
- Code List. Version. Identifier—version of a code list
- Code List. Agency. Identifier—agency that manages a code
- List—list of default agencies from DE 3055. However, *roles* defined in DE 3055 MUST NOT be used
- Code List. Agency Name.Text—name of the agency that maintains the code list.

Example*: Code List URN*

urn:dod:don:enterprise:codelist:3055:agencycode:d.02a:6:unece

The following rule describes the requirements for the xsd:schemaLocation for importing the code lists into a DON business document.

Code List Rule	
[CDL8]	The xsd:schemaLocation MUST include the complete URI used to identify the relevant code list schema.

Section 9
XML Instance Rules

9.1 Purpose

The purpose of this section is to summarize the DON XML instance rules and approach.

9.2 Root Element Rules

The DON has chosen a global element approach. In XSD, every global element is eligible to act as a root element in an instance document. The rule [RED1] says that you must declare a root element. DON business documents (XML instances) must have a single root element as defined in the corresponding DON XSD.

The root element of the XML instance must properly identify the business process or the document being expressed (see Rule [RED2]).

Example: *Sample Business Process Root Elements*

- StrikePlan,
- Order,
- Invoice,
- StockCheck

9.3 Validation

The schemaLocation attribute is a W3C schema construct that associates an XML instance document with a schema. It is used only when a schema has a target namespace. The XML instances produced and used within the DON enterprise environment must have target namespaces and must validate to a schema.

Instance Document Rule	
[IND1]	All DON instance documents MUST validate to a corresponding XSD schema.

9.4 Character Encoding

To ensure consistency within the DON, XML instance documents also have some requirements.

XML supports a wide variety of character encoding. Processors must understand which character encoding is employed in each XML document. XML 1.0 supports a default value of UTF-8 for character encoding, but best practice is to always identify the character-encoding scheme being employed.

Instance Document Rules	
[IND2]	All DON instance documents MUST always identify their character encoding within the XML declaration, except when using encryption.
[IND3]	All DON XML SHOULD be expressed using UTF-8, except when using encryption.

Example: *Character Encoding*

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

9.5 Schema Instance Namespace Declaration

The "xsi" namespace is reserved for use with instance documents and is used to define and declare four commonly used attributes:

- xsi:schemaLocation
- xsi:noNamespaceSchemaLocation
- xsi:type
- xsi:nil.

All DON XML instance documents must carry this declaration in them. This allows for use of the above attributes.

Instance Document Rule	
[IND4]	All DON instance documents MUST contain the following regular expression:"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance."

9.6 Empty Content

Usage of empty elements within XML instance documents is a source of controversy for a variety of reasons. An empty element does not simply represent missing data. It may express data that are not applicable for some reason, trigger the expression of an attribute, denote all possible values instead of just one, mark the end of a series of data, or appear as a result of an error in XML file generation.

Conversely, missing data elements can also have meaning: data not provided by a trading partner. In information exchange environments, different exchanging parties may allow, require, or ban empty elements.

In choosing to use XLink, the DON has chosen to allow empty elements in some instances. But, they are not devoid of content, they have attribute values. A totally empty element, with no attributes and no content is not allowed.

Instance Document Rule	
[IND5]	DON instance documents MUST NOT contain empty elements, except when using Xlink or KeyRef.

To ensure that no attempt is made to circumvent rule [IND5], DON also prohibits attempting to convey any tangible content or meaning by using an empty element.

Instance Document Rule	
[IND6]	An empty element MUST NOT carry meaning.

Section 10
Security Rules

Security is increasingly becoming a key focus area for XML. As XML continues to promulgate across the department, the DON will more frequently rely upon XML to successfully exchange data among organizations. The ever-increasing amount of DON XML data traffic poses a new potential vulnerability that the DON NDR must address. XML communications are typically "plain-text" or "in the clear," which denotes that the XML data is unencrypted and free for any network intermediary to exploit. The W3C and OASIS standards bodies have addressed this vulnerability with a series of formal standard recommendations:

- XML Encryption Syntax and Processing, W3C Recommendation 10, December 2002
- Decryption Transform for XML Signature, W3C Recommendation 10, December 2002
- XML-Signature Syntax and Processing, W3C Recommendation, 12 February 2002
- Security Assertion Markup Language (SAML) v1.1, OASIS Recommendation, August 2003
- Extensible Access Control Markup Language (XACML) v1.0, OASIS Recommendation.

Service-oriented architectures (SOAs) are a shift from the traditional n-tiered architecture that is considered best practice today. N-tiered architectures typically focus on perimeter security rather than the specific application requirements. Enclave security is considered mature and, as a general practice, utilizes DMZ, intrusion detection, and firewall technologies to protect their specific user communities. Increasingly, however, the applications that users typically access are spread across many different organizational enclaves. Thus, the DON must adapt to the latest SOA security enabling technologies.

Web Services and its associated technologies such as SOAP, WSDL, UDDI, and WS-Interoperability are out of the scope of this security discussion. One emerging standard, currently in draft status, is WS-Security.[1] This WS-Security draft standard warrants a brief mention in this section because it addresses web services, specifically SOAP-level messaging, as a framework. This SOAP-level messaging framework potentially can provide a methodology for encrypted XML data transfer as well as data integrity and nonrepudiation via digital signature.

[1] WS-Security, Web Services Security, has been proposed by an industry consortium (IBM, Microsoft, and Verisign) and has been submitted for ratification by OASIS.

10.1 Purpose

The purpose of this section is to summarize the DON XML security rules and approach.

10.2 Security Rules

The DON NDR has adopted the W3C and OASIS XML security standard recommendations as the DON standard for securing and signing XML communications.

Security Rule	
[SEC1]	W3C and OASIS recommendations that are applicable to XML security and digital signing MUST be used where appropriate.

The DON NDR addresses the following two main focus areas for securing XML communications:

- Use of the XML Encryption Syntax and Processing standard (XMLENC)
- Use of the XML Digital Signature Syntax and Processing (XMLDSIG).

The W3C XMLDSIG standard (e.g., recommendation) details the specification for digitally signing XML components using either symmetric (e.g., AES, DES) or asymmetric (e.g., RSA) key cryptography.[2] A digitally signed XML component ensures the integrity of that component. Similarly, a signed XML document ensures the data integrity of that document.

Security Rule	
[SEC2]	W3C XMLDSIG MUST be used to digitally sign XML components where appropriate.

[2] According to *XML-Signature Syntax and Processing, W3C Recommendation, 12 February 2002*, "XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere."

Two examples of the XMLDSIG syntax for encrypting an XML element and document are indicated below:

Example: *Digitally Signed XML Document*

```
<Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
        c14n-20010315"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-
            sha1"/>
                <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
            <Transforms3>
                <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
                20010315"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
    </Reference>
</SignedInfo>
<SignatureValue>MC0CFFrVLtRlk=…</SignatureValue>
    <KeyInfo>
        <KeyValue>
            <DSAKeyValue>
            <P>…</P><Q>…</Q><G>…</G><Y>…</Y>
            </DSAKeyValue>
        </KeyValue>
    </KeyInfo>
</Signature>
```

The <Transforms> usage allows operations to occur prior to digesting the data, such as when XPath is used to derive XML components from another document. Consequently, those XML components that are excluded in the derived document can change without affecting the signature validity of the digitally signed document. If transforms is not used, then the XML component is signed immediately without following any prescribed operations.

Example: *XML Schema Syntax Supporting a Digitally Signed Instance*

```
<schema targetNamespace="urn:us:gov:dod:don:enterprise:strikeplan:1:0"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" elementFormDefaut="qualified">
```

[3] According to *XML-Signature Syntax and Processing, W3C Recommendation, 12 February 2002,* "Transforms can include operations such as canonicalization, encoding/decoding (including compression/inflation), XSLT, XPath, XML schema validation, or XInclude. XPath transforms permit the signer to derive an XML document that omits portions of the source document."

The W3C XMLENC standard details the specification for digitally encrypting XML components using either symmetric or asymmetric key cryptography. A digitally encrypted XML component ensures the persistent data confidentiality of that component. Similarly, an encrypted XML document ensures the data confidentiality of that document.

Security Rule	
[SEC3]	W3C XMLENC MUST be used to digitally encrypt XML components where appropriate.

Two examples of the XMLENC syntax for encrypting an XML element and document are indicated below:

Example: *Encrypting an XML Element*

```
<?xml version="1.0"?>
    <StrikePlan xmlns="urn:us:gov:dod:don:enterprise:strikeplan:1:0">
      <Name>John Smith</Name>
        <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
      xmlns="http://www.w3.org/2001/04/xmlenc#">
    <CipherData>
      <CipherValue>A23999B45C5F6</CipherValue>
    </CipherData>
</EncryptedData>
</StrikePlan>
```

It is important to note that the XMLENC rules do *not* apply to non-persistent (session) encryption technologies like Secure Socket Layer/Transport Layer Security (SSL/TLS). For many organizations, SSL/TLS provides adequate data confidentiality for XML data transfer. If the session-level encryption does not meet the level of data confidentiality prescribed by the organization, then data-level encryption (XMLENC) must be used where appropriate.

Example: *Encrypting an XML Document*

```
<?xml version="1.0"?>
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
    MimeType="text/xml">
    <CipherData>
      <CipherValue>A2355DB45C56</CipherValue>
    </CipherData>
</EncryptedData>
```

Appendix A
Rules Tables

This appendix contains the following rules tables:

Attribute Declaration Rules	
[ATD1]	User-defined attributes SHOULD NOT be used.
[ATD2]	If a DON `xsd:SchemaExpression` contains one or more common attributes that apply to all DON elements contained or included or imported therein, the common attributes MUST be declared as part of a global attribute group.
[ATD3]	An objectIDREF xsd:attribute MUST be declared globally in the DON Enterprise BIE Reusable Schema module.
[ATD4]	The objectIDRef xsd:attribute value MUST be equal to the value of an ID ccts:BBIE element.

Code List Rules	
[CDL1]	All codes used in DON MUST be part of a DON-maintained or externally maintained Code List Schema module.
[CDL2]	The DON library MAY design and use an internal code list if an existing external code list needs to be extended or if no suitable external code list exists.
[CDL3]	All DON-maintained or used code lists MUST be enumerated using the DON Code List Schema module.
[CDL4]	The name of each DON Code List Schema module MUST be: <Owning Organization>-Codelist-<Code List Name>- <Version>.
[CDL5]	An xsd:Import element MUST be declared for every code list required in a DON schema.
[CDL6]	Users of the DON library MAY identify any subset they wish from an identified code list for their own trading community conformance requirements.
[CDL7]	The namespace name of each DON Code List Schema module MUST conform to the following pattern: urn:dod:don:enterprise:codeList:<Code List.Identification.Identifier>:<Code List.Name.Text>:<Code List.Version.Identifier>:<Code List.Agency.Identifier>:<Code List. AgencyName.Text>.
[CDL8]	The xsd:schemaLocation MUST include the complete URI used to identify the relevant code list schema.

Complex Type Definition Rules	
[CTD1]	Every ccts:ABIE xsd:complexType MUST use xsd:sequence or xsd:choice with global element references and/or local element declarations to reflect each property of its class.
[CTD2]	The xsd:any MAY be used in the content model of an DON ABIE type when the ABIE represents an object defined by an external XML business standard approved by the DON BSC.
[CTD3]	A named xsd:complexType MUST be defined for every ABIE and BBIE Property identified in a DON information model.
[CTD4]	Every BBIE property MUST define a named xsd:complexType.
[CTD5]	Every ccts:BBIE Property complexType definition must be based on a complex type representing either a ccts:QualifiedDataType or ccts:UnqualifedDataType.
[CTD6]	Every ccts:BBIE Property complex type content model MUST use xsd:simpleContent.
[CTD7]	Every ccts:BBIE Property xsd:simpleContent element MUST contain either the xsd:extension or xsd:restriction element.
[CTD8]	Code and ID ccts:BBIE Property complex types MAY use the xsd:choice element to reference global elements defined in standardized ID Scheme or Code List Schema modules.

[CTD9]	EVERY primary and secondary representation term MUST have an xsd:complexType defined in the DON Enterprise Unqualified ccts:DataTypes Schema module (UDT)
[CTD10]	Every ccts:DataType complex type MUST use xsd:simpleContent.
[CTD11]	Every ccts:DataType SHOULD use xsd:restriction.
[CTD12]	Every Qualified ccts:DataType MUST use the xsd:base attribute to define another ccts:DataType.
[CTD13]	Each ccts:SupplementaryComponent xsd:attribute user-defined xsd:simpleType MUST only be used when the ccts:SupplementaryComponent value is based on a standardized code list for which a BSC approved has been created.
[CTD14]	Every Qualified ccts:DataType (QDT) xsd:complexType content model xsd:restriction element MUST use the xsd:base attribute to define the basis as a UDT xsd:complexType.
[CTD15]	Each ccts:SupplementaryComponent xsd:attribute “use” MUST define the occurrence as either “required” or “optional.”

Complex Type Naming Rules	
[CTN1]	A DON xsd:complexType name based on a ccts:ABIE MUST be the ccts:DictionaryEntryName object class, property term, any qualifiers and the representation term with the separators removed and the “Details” suffix replaced with “Type.”
[CTN2]	A DON xsd:complexType name based on a ccts:BBIE Property MUST, at a minimum, include the ccts:DictionaryEntryName property term, any qualifiers and the representation term, with the separators removed and with the “Type” suffix appended after the representation term.
[CTN3]	A DON xsd:complexType name based on a ccts:QualifiedDataType (QDT) MUST, at a minimum, include the dictionary entry name of the ccts:UnualifiedDataType (UDT) it derives from, with the separators removed.
[CTN4]	A DON ccts:DataType name MUST be the ccts:DictionaryEntryName with the separators removed.

Documentation Rules	
[DOC1]	Every data type definition MUST contain a structured set of annotations in the following sequence and pattern: ■ UniqueIdentifier (mandatory): The identifier that references a data type instance in a unique and unambiguous way. ■ CategoryCode (mandatory): The category to which the object belongs. For example, BBIE, ABIE, ASBIE. ■ DictionaryEntryName (mandatory): The official name of a data type. ■ Definition (mandatory): The semantic meaning of a data type. ■ Version (mandatory): An indication of the evolution over time of a data type instance. ■ QualifierObjectClass (optional): The qualifier for the object class. ■ ObjectClass: The object class represented by the data type. ■ Qualifier Term (mandatory): A semantically meaningful name that differentiates the data type from its underlying UDT. ■ Usage Rule (optional, repetitive): A constraint that describes specific conditions that are applicable to the data type.
[DOC2]	A data type definition MAY contain one or more content component restrictions to provide additional information on the relationship between the data type and its corresponding UDT. If used, the content component restrictions must contain a structured set of annotations in the following patterns: ■ RestrictionType (mandatory): Defines the type of format restriction that applies to the content component. ■ RestrictionValue (mandatory): The actual value of the format restriction that applies to the content component. ■ ExpressionType (optional): Defines the type of the regular expression of the restriction value.
[DOC3]	A data type definition MAY contain one or more supplementary component restrictions to provide additional information on the relationship between the data type and its corresponding UDT. If used, the supplementary component restrictions must contain a structured set of annotations in the following patterns: ■ SupplementaryComponentName (mandatory): Identifies the supplementary component on which the restriction applies. ■ RestrictionValue (mandatory, repetitive): The actual value(s) that is (are) valid for the supplementary component.

[DOC4]	Every Basic Business Information Entity (BBIE) definition MUST contain a structured set of annotations in the following patterns: ■ Unique Identifier (mandatory): The identifier that references a BBIE instance in a unique and unambiguous way. ■ CategoryCode (mandatory): The category to which the object belongs. In this case, the value will always be BBIE. ■ Dictionary Entry Name (mandatory): The official name of a BBIE. ■ Version (mandatory): An indication of the evolution over time of a BBIE instance. ■ Definition (mandatory): The semantic meaning of a BBIE. ■ Cardinality (mandatory): Indication whether the BBIE property represents a not-applicable, optional, mandatory, and/or repetitive characteristic of the Aggregate Business Information Entity. ■ QualifierTerm (optional): Qualifies the property term of the associated core component property in the associated Aggregate Core Component. ■ UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the BBIE. ■ ConstraintLanguage (optional, repetitive): A formal description of a way the BBIE is derived from the corresponding stored core component and stored business context. ■ BusinessTerm (optional, repetitive): A synonym by which the BBIE is commonly known and used in the business. ■ Example (optional, repetitive): Example of a possible value of a BBIE.

[DOC5]	Every Aggregate Business Information Entity (ABIE) definition MUST contain a structured set of annotations in the following patterns: ■ UniqueIdentifier (mandatory): The identifier that references an ABIE instance in a unique and unambiguous way. ■ CategoryCode (mandatory): The category to which the object belongs. In this case, the value will always be ABIE. ■ Version (mandatory): An indication of the evolution over time of an ABIE instance. ■ DictionaryEntryName (mandatory): The official name of an ABIE. ■ Definition (mandatory): The semantic meaning of an ABIE. ■ QualifierTerm (mandatory): Qualifies the object class term of the associated Aggregate Core Component. ■ UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the ABIE. ■ ConstraintLanguage (optional, repetitive): A formal description of a way the ABIE is derived from the corresponding stored core component and stored business context. ■ BusinessTerm (optional, repetitive): A synonym by which the ABIE is commonly known and used in the business. ■ Example (optional, repetitive): Example of a possible value of an ABIE.

[DOC6]	Every Association Business Information Entity (ASBIE) definition MUST contain a structured set of annotations in the following patterns: ■ UniqueIdentifier (mandatory): The identifier that references an ASBIE instance in a unique and unambiguous way. ■ CategoryCode (mandatory): The category to which the object belongs. In this case, the value will always be ASBIE. ■ DictionaryEntryName (mandatory): The official name of an ASBIE. ■ Definition (mandatory): The semantic meaning of an ASBIE. ■ Version (mandatory): An indication of the evolution over time of an ASBIE. ■ Cardinality (mandatory): Indication whether the ASBIE property represents a not-applicable, optional, mandatory, and/or repetitive characteristic of the ABIE. ■ QualifierTerm (optional): Qualifies the property term of the associated core component property in the associated aggregate core component. ■ UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the ASBIE. ■ ConstraintLanguage (optional, repetitive): A formal description of a way the ASBIE is derived from the corresponding stored core component and stored business context. ■ BusinessTerm (optional, repetitive): A synonym term under which the ASBIE is commonly known and used in the business. ■ Example (optional, repetitive): Example of a possible value of an ASBIE
[DOC7]	Every Core Component definition MUST contain a structured set of annotations in the following patterns: ■ UniqueIdentifier (mandatory): The identifier that references a Core Component instance in a unique and unambiguous way. ■ CategoryCode (mandatory): The category to which the object belongs. In this case the value will always be CCT. ■ DictionaryEntryName (mandatory): The official name of a Core Component. ■ Definition (mandatory): The semantic meaning of a Core Component. ■ ObjectClass: The object class represented by the type. ■ PropertyTerm: The property term represented by the type. ■ Version (mandatory): An indication of the evolution over time of a Core Component instance. ■ Usage Rule (optional, repetitive): A constraint that describes specific conditions that are applicable to the BBIE. ■ Business Term (optional, repetitive): A synonym by which the BBIE is commonly known and used in the business.

[DOC8]	Every element declaration MUST contain an annotation as follows: <Documentation>[Dictionary Entry Name]</Documentation> where Dictionary Entry Name is the complete name (not the tag name) that is the unique official name of the element in the DON library.
[DOC9]	For each DON construct containing a code, the DON documentation MUST identify the zero or more code lists that MUST be minimally supported when the construct is used: ■ Prefix (mandatory): The code prefix, for example “cnt” for country code list. ■ CodeListQualifier (mandatory): The qualifier for the code list, for example, “ISO 3166-1.” ■ CodeListAgency: The maintainer of the code list, for example “6.” ■ CodeListVersion: The version of the code list, for example “0.3.

Element Declaration Rules	
[ELD1]	All element declarations MUST be global with the exception of `Identifiers, Measures, and Codes`, which MAY be declared as local elements if, and only if, approved by the FNC and BSC
[ELD2]	For every xsd:complexType representing a CCTS:ABIE, a global element MUST be declared.
[ELD3]	For every xsd:complexType representing a CCTS:BBIE Property, a global element MUST be declared.
[ELD4]	For each CCTS:ASBIE, a global element MUST be declared.
[ELD5]	For CCTS:BBIEs that are based on ID, code, and measure, a local element MAY be declared in the xsd:complexType of the parent ABIE.
[ELD6]	Empty elements SHALL NOT be declared except for reference elements and Xlink elements, which MUST be approved by the cognizant FNC and BSC.

Element Naming Rules	
[ELN1]	A DON CCTS:ABIE element name MUST be the same as the corresponding xsd:complexType to which it is bound, with the word “Type” removed.
[ELN2]	A DON global element based on a CCTS:BBIE Property element MUST be the same name of its corresponding xsd:complexType with the word “Type” removed, unless the object class can be used for semantic clarity.
[ELN3]	A DON CCTS:ASBIE name MUST be based on the CCTS:ASBIE dictionary entry and MAY contain an optional suffix to provide clear cardinality.
[ELN4]	Each root element in a DON Root Schema module document MUST be named according to the portion of the business process initiated or the content item published.

General Naming Rules	
[GNR1]	DON XML element, attribute, and type names MUST be in the English language, using the Oxford English Dictionary for Writers and Editors (Latest Ed.). Where both American and English spellings of the same word are provided, the American spelling MUST be used.
[GNR2]	DON XML element, attribute, and type names MUST conform to CCTS dictionary entry names with all separators and spaces removed.
[GNR3]	DON enterprise XML element, attribute, and type names MUST NOT use abbreviations or other word truncations (e.g. acronyms), except those in the approved list published by the cognizant FNC.
[GNR4]	Abbreviations and acronyms MUST be submitted to an FNC for approval.
[GNR5]	The abbreviations and acronyms list approved by the BSC and FNC MUST be used.
[GNR6]	DON XML element, attribute, and type names MUST be in singular form unless the concept itself is plural (example: goods).
[GNR7]	The UpperCamelCase (UCC) convention MUST be used for naming elements and types.
[GNR8]	The lowerCamelCase (LCC) convention MUST be used for naming attributes.
[GNR9]	The xsd:unique identity constraints names MUST be the same as the object class of the ABIE being identified uniquely plus the suffix “Key.”
[GNR10]	The xsd:keyref identity constraint names MUST be consist of the name of the referencing object class plus the name of the referenced object class plus the suffix “REFKey.”

General XSD Rules	
[GXS1]	The root element in all DON Schema modules MUST contain the following declaration: “xmlns:xsd=http://www.w3.org/2001/XMLSchema.”
[GXS2]	DON schema developers MAY provide a run-time schema devoid of documentation in addition to the fully annotated version.
[GXS3]	DON UDT built-in simple types MUST be used wherever possible.
[GXS4]	The xsd:substitution groups feature SHOULD NOT be used.
[GXS5]	The xsd:final attribute MUST be used on xsd:complexType definitions derived by restriction to prevent further restriction or extensions.
[GXS6]	xsd:notations MUST NOT be used.
[GXS7]	The xsd:all element MUST NOT be used.
[GXS8]	The xsd:choice element MAY be used.
[GXS9]	The xsd:namespace attribute MUST be defined when using the xsd:any element and MUST have a URI value representing the namespace of a BSC-approved XML vocabulary.
[GXS10]	The xsd built-in nillable attribute MUST NOT be used.

[GXS11]	DON schema SHOULD NOT use xsd:appinfo. If used, xsd:appinfo MUST only be used to convey non-normative information.
[GXS12]	Complex type extension or restriction MAY be used.
[GXS13]	Any xsd:complexTypes derived by restriction MUST NOT be further extended.
[GXS14]	The code list xsd:import element MUST contain the namespace and schema location attributes.
[GXS15]	The xsd:include feature MUST be used only when applicable
[GXS16]	The xsd:union technique MAY be used for code lists.

Instance Document Rules	
[IND1]	All DON instance documents MUST validate to a corresponding XSD schema.
[IND2]	All DON instance documents MUST always identify their character encoding within the XML declaration, except when using encryption.
[IND3]	All DON XML SHOULD be expressed using UTF-8, except when using encryption.
[IND4]	All DON instance documents MUST contain the following regular expression: "xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance."
[IND5]	DON instance documents MUST NOT contain empty elements, except when using Xlink or KeyRef.
[IND6]	An empty element MUST NOT carry meaning.

Information Analysis Rules	
[INF1]	A qualified ccts:DataType must be created in the Syntax Neutral Model for each BBIE that uses domain restriction.

Modeling Constraints Rules	
[MDC1]	DON XML components MUST be based on approved ccts:UnqualifiedDataTypes that are based on ccts:CoreComponentTypes.
[MDC2]	DON information models MUST define classes based on Aggregate Business Information Entities (ABIEs) and ccts:DataTypes.
[MDC3]	If a DON ccts:DataType is extended or restricted, it MUST retain the original business context.
[MDC4]	Mixed content MAY only be used when an XML schema component is defined by a namespace from a BSC-approved business standard (e.g., XHTML).

Namespace Rules	
[NMS1]	All DON enterprise and development namespaces MUST use the base URN "urn:us:gov:dod:don."
[NMS2]	URNs MUST be in lowercase, except multiple words, which MUST use lower camel case and the root element, which will use upper camel case.
[NMS3]	All Business Standards Council (BSC) approved enterprise (root) schema MUST reside in the DON enterprise root namespace.
[NMS4]	The namespace for DON root schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:schema:<root name>:<major>:<minor>.
[NMS5]	The namespace for DON QDT schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:qdt: <major>:<minor>.
[NMS6]	The namespace for DON UDT schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:udt: <major>:<minor>.
[NMS7]	The namespace for DON Code List schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:<codeListName>:<major>:<minor>.
[NMS8]	The namespace for DON Identifier schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:<identifierName>:<major>:<minor>.
[NMS9]	The namespace for DON Unqualified Code List and Identifier schema holding enterprise status MUST be of this form: urn:us:gov:dod:don:enterprise:UCodeIdentifer: <major>:<minor>.
[NMS10]	The namespace for DON schema holding development status MUST be of this form: urn:us:gov:dod:don:<service branch>:<FAM>:<command>:<component>:<root name>:<major>:<minor>
[NMS11]	All DON namespace declarations MUST be qualified.
[NMS12]	DON enterprise schema MUST declare a namespace using the xsd:targetNamespace attribute.
[NMS13]	The DON enterprise namespace MUST contain only DON-developed schema modules.
[NMS14]	Each DON:CodeList schema module MUST be maintained in a separate namespace.

Root Element Declaration Rules	
[RED1]	Each DON Root-Level Schema module MUST identify at least one global element declaration that defines the content in the schema expression. That global element MUST include an xsd:annotation child element, which MUST further contain an xsd:documentation child element that declares the following: “This element MUST be conveyed as the root element in any instance document based on this schema expression.”
[RED2]	Every root element in a DON document MUST be named according to the portion of the business process or document it initiates.

Security Rules	
[SEC1]	W3C and OASIS recommendations that are applicable to XML security and digital signing MUST be used where appropriate.
[SEC2]	W3C XMLDSIG MUST be used to digitally sign XML components where appropriate.
[SEC3]	W3C XMLENC MUST be used to digitally encrypt XML components where appropriate.

Schema Structure Modularity Rules	
[SSM1]	Enterprise level root schema MAY import external qualified data types, core components, code list, and identifier schema that have been approved by the BSC and cognizant FNC authority.
[SSM2]	Every DON enterprise root schema MUST import the DON Enterprise Reusable BIE schema (DON-Enterprise-Reusable-z.z.z.xsd).
[SSM3]	All enterprise global elements and named complex types representing ccts:Components must reside in the Enterprise BIE Reusable Schema module.
[SSM4]	All DON development root schemas MUST be submitted to both the cognizant FNC and the appropriate namespace in the DoD registry.
[SSM5]	Development global elements and named complex types MUST reside in a no namespace schema that is included in the root schema.
[SSM6]	Development qualified data types (complex types) MUST reside in a no namespace qualified data types schema that is included in the root schema submission.
[SSM7]	A run-time schema MAY be created to meet performance requirements of the application in the run-time environment.
[SSM8]	All modifications, updates, revisions, and new releases MUST first go through the DON schema approval process before the changes can be incorporated into the run-time schema.
[SSM9]	Xsd:Include MAY only be used in a Development or enterprise run-time root schema

Standards Adherence Rules	
[STA1]	All DON XSD schema design MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.
[STA2]	All DON schema and messages MUST be based on the W3C suite of technical specifications holding recommendation status.

Versioning Rules	
[VER1]	XML schema version information MUST be defined in a schema and match the version of the namespace in which it resides.
[VER2]	The major-version field MUST equal "1" for the first release of a namespace.
[VER3]	The major-version field of a namespace or schema MUST be incremented when the proposed Schema changes impact the compatibility of any previous XML instance based on the related Schema.
[VER4]	The minor-version field of a namespace or schema MUST be incremented if all XML instances will continue to validate successfully with the new version of the schema.
[VER5]	Every major- or minor-version enterprise DON schema MUST include the version information of: <major>:<minor> of the namespace name.
[VER6]	The first schema module to appear in a DON Enterprise BIE or Qualified Data Type namespace MUST have a version attribute equal to the following: <major-number>:<minor-number>:<schema version number>.
[VER7]	Schema in the DON Enterprise BIE or Qualified Data Type namespaces MUST increment the third digit of the schema version attribute when a schema is changed.
[VER8]	Minor version changes to enterprise, qualified data types, and root schema components MUST be backward compatible and validate all previous versions of XML instances.
[VER9]	New versions of the schema created in the DON Enterprise BIE or Qualified Data Types namespace MUST *include* the previous schema version.
[VER10]	New core component versions MUST not alter the component's ccts:DictionaryEntryName or definition.

Appendix B
Bibliography

DON CIO Memorandum, Interim Policy on the Use of Extensible Markup Language (XML) for Data Exchange, 6 September 2001.

SECNAVINST 5000.36, Data Management and Interoperability, 1 November 2001.

Under Secretary of the Navy Memorandum, Designation of Department of the Navy (DON) Functional Area Managers, 14 May 2002.

DoD CIO Memorandum, Policy for Registration of Extensible Markup Language (XML), 22 April 2002.

DoD Joint Technical Architecture, Version 5.0, April 4, 2003.

DOD Directive 8120.1, Life-Cycle Management (LCM) of Automated Information Systems (AISs), 14 January 1993.

DOD Directive 8100.1, Global Information Grid (GIG) Overarching Policy, 19 September 2002.

DoD 8910.1-M, DoD Procedures for Management of Information Requirements, June 1998.

DoD Directive 7740.1, Information Resources Management Program, 20 June 1983.

DoD Directive 8000.1, Management of DoD Information Resources and Information Technology, 27 February 2002.

UN/CEFACT, Core Components Technical Specification, Part 8 of the ebXML Framework, 15 November 2003, Version 2.01.

ebXML Technical Report, Core Component Overview v1.05.

ebXML Technical Report, Core Component Discovery and Analysis v1.04.

ebXML Technical Report, Context and Re-Usability of Core Components v1.04.

ebXML Technical Report, Guide to the Core Components Dictionary v1.04.

ebXML Technical Report, Naming Convention for Core Components v1.04.

ebXML Technical Report, Document Assembly and Context Rules v1.04.

ebXML Technical Report, Catalogue of Context Categories v1.04.

ebXML Technical Report, Core Component Dictionary v1.04.

ebXML Technical Report, Core Component Structure v1.04.

Information Technology Metadata Registries: Framework for the Specification and Standardization of Data Elements, International Standardization Organization, ISO 11179-1.

Information Technology Metadata Registries: Classification of Concepts for the Identification of Domains, International Standardization Organization, ISO 11179-2.

Information Technology Metadata Registries: Registry Metamodel, International Standardization Organization, ISO 11179-3.

Information Technology Metadata Registries: Rules and Guidelines for the Formulation of Data Definitions, International Standardization Organization, ISO 11179-4.

Appendix C
Glossary/Acronyms

AES	Advanced Encryption Standard
Aggregate Business Information Entity (ABIE)	A collection of related pieces of business information that together convey a distinct business meaning in a specific business context. Expressed in modeling terms, it is the representation of an object class, in a specific business context.
Aggregate Core Component (ACC)	A collection of related pieces of business information that together convey a distinct business meaning, independent of any specific business context. Expressed in modeling terms, it is the representation of an object class, independent of any specific business context.
Association Business Information Entity (ASBIE)	A business information entity that represents a complex business characteristic of a specific object class in a specific business context. It has a unique business semantic definition. An ASBIE represents an association business information entity property and is associated to an ABIE, which describes its structure. An ABIE is derived from an association core component.
Association Core Component (ASCC)	A core component that constitutes a complex business characteristic of a specific aggregate core component that represents an object class. It has a unique business semantic definition. An ASCC represents an association core component property and is associated to an ACC, which describes its structure.
Basic Business Information Entity (BBIE)	A business information entity that represents a singular business characteristic of a specific object class in a specific business context. It has a unique business semantic definition. A BBIE represents a basic business information entity property and is therefore linked to a data type, which describes it values. A BBIE is derived from a basic core component.

Basic Core Component (BCC)	A core component that constitutes a singular business characteristic of a specific aggregate core component that represents an object class. It has a unique business semantic definition. A BCC represents a basic core component property and is therefore of a data type, which defines its set of values. BCCs function as the properties of aggregate core components.
BBIE ComplexType	An xsd:complexType based on a BBIE where a BBIE property complex type cannot be defined. BBIE ComplexType names always contain an object class.
BBIE Global Element	A global element whose type is a BBIE ComplexType. BBIE global element names always contain an object class.
BBIE Property ComplexType	An xsd:complexType defined based on the existence of a unique Property Term/Representation Term pair in a DON model.
BBIE Property Global Element	A global element whose type is a BBIE Property ComplexType.
BSC	Business Standards Council
Business Context	The formal description of a specific business circumstance as identified by the values of a set of context categories, allowing different business circumstances to be uniquely distinguished.
Business Information Entity (BIE)	A piece of business data or a group of pieces of business data with a unique business semantic definition. A BIE can be a BBIE, an ASBIE, or an ABIE.
CCTS	Core Component Technical Specification
CDP	Core Documentation Parameter
Content DataType	A qualified ccts:DataType based on the “Text.Type” whose contents are restricted by some document or presentation oriented XML vocabulary such as XHTML.
Core Component Type (CCT)	A core component that consists of one and only one content component and carries the actual content plus one or more supplementary components giving an essential extra definition to the content component. CCTs do not have business semantics.

Data Type (DT)	The set of valid values that can be used for a particular basic core component property or basic business information entity property. It is defined by specifying restrictions on the core component type that forms the basis of the data type.
DES	Digital Encryption Standard
DON	Department of Navy
DRM	Data Reference Model
DTD	Document Type Definition
ELD	Element declaration
ELN	Element naming
FDM	Functional Data Manager
FEA	Federal Enterprise Architecture
FNC	Functional Namespace Coordinator
GXS	General XML Schema Rules
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
IT	Information Technology
Local Reference Element	A local element declared in the content model of a global element representing an ASBIE. Local reference elements are only declared for ASBIEs representing reference associations that are one-to-many. They are declared as empty and contain a single objectIDREF attribute.

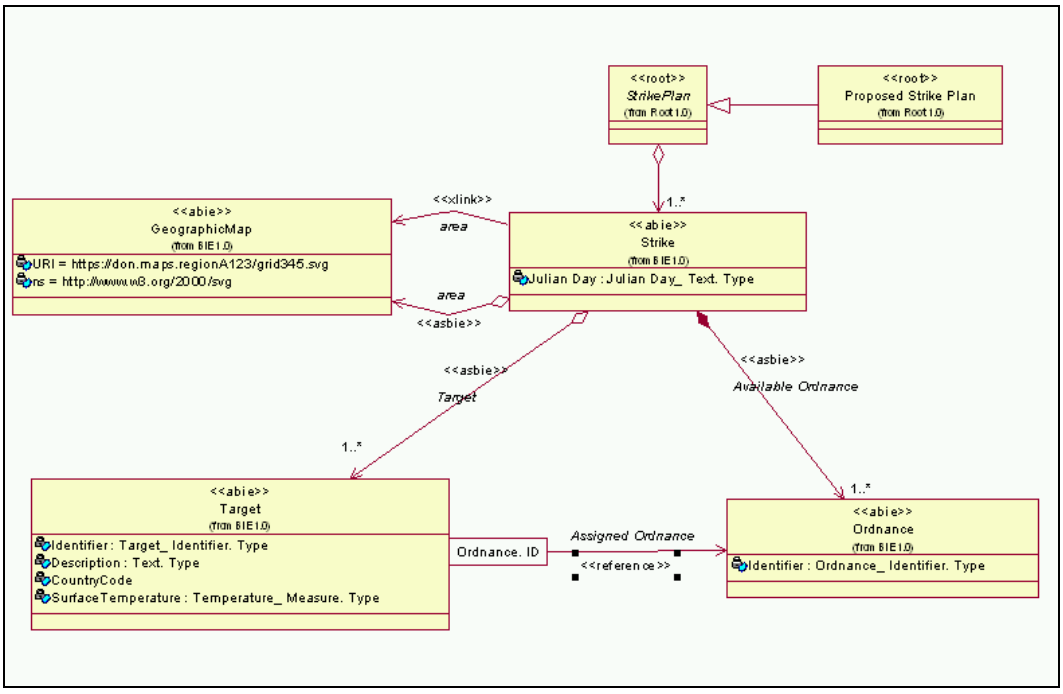
Naming Convention	The set of rules that together constitute how the dictionary entry name for core components and business information entities are constructed.
NDR	naming and design rules
NID	namespace identifier
NSS	namespace-specific string
OASIS	Organization for the Advancement of Structured Information Systems
Object Identifier Reference Attribute (objectIDREF)	An attribute with a value equal to another XML element representing an identifier BBIE.
Qualified DataType (QDT)	A ccts:DataType that is based on another ccts:DataType. The two are differentiated by the addition of a qualifier term to the former's dictionary name.
Qualified Datatypes Schema Module	A Schema module that defines named complexTypes based on ccts:DataTypes that are qualified.
Reference Associations	Associations between model elements that are not strong or parent/child aggregations. Reference associations may be expressed in XML by normalizing instance via keyref/unique identity constraints.
RFC	request for comments
RSA	Rivest, Shamir, & Adleman (public key encryption technology)
SAML	Security Assertion Markup Language
SOA	Service-oriented architectures
SSL	Secure Socket Layer
SVG	scalable vector graphic
TAT	Technical Assistance Team
TC	Technical Committee
TLS	Transport Layer Security
UBL	Universal Business Language

UDDI	Universal Data Description Interface
UDT	Unqualified Datatypes
UML	Unified Modeling Language
UMM	Modeling Methodology
Unqualified Datatypes Schema Module	A Schema module that defines named complexTypes based upon the ccts:PrimaryRepresentationTerms and ccts:SecondaryRepresentationTerms. These ccts:DataTypes are unqualified and serve as the basis for all qualified datatypes.
URI	Uniform Resource Identifiers
URN	Uniform Resource Name
URN Version	The version number in the namespace of a URN. This version is the last two fields of the URN (e.g., bie:1:0)
VCS	voluntary consensus standards
Version	An indication of the evolution over time of an instance of a core component, data type, business context, or business information entity.
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XACML	Extensible Access Control Markup Language
XHTML	Extensible Hypertext Markup Language
XLink	Specifies constructs that may be inserted into XML resources to describe links between objects. It uses XML syntax to create structures that can describe the simple unidirectional hyperlinks of today's HTML as well as more sophisticated multi-ended and typed links.
XML	Extensible Markup Language
XMLDSIG	XML Digital Signature Syntax and Processing
XMLENC	XML Encryption Syntax and Processing standard
XSD Version	Refers to the version number in the XSD Schema; this is an attribute of xsd:schema.
XSLT	extensible stylesheet language transformation

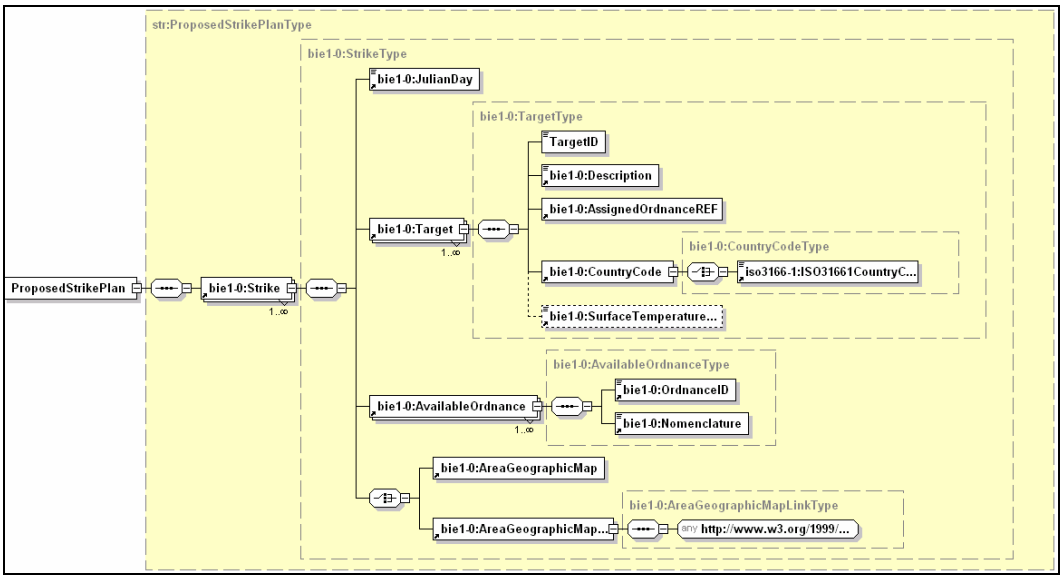
Appendix D
Schema Examples

1.1 Enterprise Namespace Example: Strike Plan

1.1.1 StrikePlan UML Object Model



1.1.2 StrikePlan Schema Model



1.1.3 DON-StrikePlan.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ProposedStrikePlan xmlns="urn:us:gov:dod:don:enterprise:schema:StrikePlan:1:0" xmlns:bie1-
0="urn:us:gov:dod:don:enterprise:bie:1:0" xmlns:iso3166-1="urn:iso:codelist:3166-
1:CountryIdentificationCode:1:0" xmlns:qdt1-0="urn:us:gov:dod:don:enterprise:qdt:1:0" xmlns:udt1-
0="urn:us:gov:dod:don:enterprise:udt:1:0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:us:gov:dod:don:enterprise:schema:StrikePlan:1:0
DON-Enterprise-StrikePlanRoot-1.0.0.xsd">
    <bie1-0:Strike>
        <bie1-0:JulianDay>0000</bie1-0:JulianDay>
        <bie1-0:Target>
            <bie1-0:TargetID>A0A000</bie1-0:TargetID>
            <bie1-0:Description>String</bie1-0:Description>
            <bie1-0:AssignedOrdnanceREF bie1-0:objectIDREF="A00001"/>
            <bie1-0:CountryCode>
                <iso3166-1:ISO31661CountryCode>US</iso3166-1:ISO31661CountryCode>
            </bie1-0:CountryCode>
            <bie1-0:SurfaceTemperatureMeasure>3.14</bie1-0:SurfaceTemperatureMeasure>
        </bie1-0:Target>
        <bie1-0:AvailableOrdnance>
            <bie1-0:OrdnanceID>A00000</bie1-0:OrdnanceID>
            <bie1-0:Nomenclature>String</bie1-0:Nomenclature>
        </bie1-0:AvailableOrdnance>
        <bie1-0:AvailableOrdnance>
            <bie1-0:OrdnanceID>A00001</bie1-0:OrdnanceID>
            <bie1-0:Nomenclature>String</bie1-0:Nomenclature>
        </bie1-0:AvailableOrdnance>
        <bie1-0:AreaGeographicMapLink xlink:href="https://don.maps.regionA123/grid345.svg"/>
    </bie1-0:Strike>
</ProposedStrikePlan>
```

1.1.4 DON-Enterprise-StrikePlanRoot-1.0.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Department of the Navy (DON) XML Naming and Design Rules (NDRs)
  DocType - DON Enterprise Strike Plan Schema
  Purpose - Defines the structure of all messages involved in the Strike business process.

  Version - 1.0.0
  Version Description - Initial releas of the Schema.
  Date - 3/31/2004

  This XML Schema is provided as an example only for purposes of illustrating applications of the
DON XML NDRs Version 2.0.
 -->
<xsd:schema targetNamespace="urn:us:gov:dod:don:enterprise:schema:StrikePlan:1:0"
xmlns:str="urn:us:gov:dod:don:enterprise:schema:StrikePlan:1:0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:bie1-
0="urn:us:gov:dod:don:enterprise:bie:1:0" xmlns:qdt1-0="urn:us:gov:dod:don:enterprise:qdt:1:0"
xmlns:udt1-0="urn:us:gov:dod:don:enterprise:udt:1:0" xmlns:iso3166-1="urn:iso:codelist:3166-
1:CountryIdentificationCode:1:0" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0.0">
    <xsd:import namespace="urn:us:gov:dod:don:enterprise:bie:1:0" schemaLocation="DON-
Enterprise-BIE-Reusable-1.0.0.xsd"/>
    <xsd:element name="ProposedStrikePlan" type="str:ProposedStrikePlanType">
        <xsd:annotation>
            <xsd:documentation>This element MUST be conveyed as the root element in any
instance document based on this schema expression.</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
    <xsd:complexType name="ProposedStrikePlanType">
        <xsd:complexContent>
            <xsd:extension base="str:StrikePlanType"/>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="StrikePlanType" abstract="true">
        <xsd:sequence>
            <xsd:element ref="bie1-0:Strike" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

1.1.5 DON-Enterprise-BIE-Reusable-1.0.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Department of the Navy (DON) XML Naming and Design Rules (NDRs)
  DocType - DON Enterprise Business Information Entity (BIE)and Reusable Schema
  Purpose - defines and declares reusable XML Schema constructs for all BIEs that have been
approved in the DON Enterprise namespace.

  Version - 1.0.0
  Version Description - Initial release of Schema Module
  Date - 3/31/2004

  This XML Schema is provided as an example only for purposes of illustrating applications of the
DON XML NDRs Version 2.0.
 -->
<xsd:schema targetNamespace="urn:us:gov:dod:don:enterprise:bie:1:0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:tmuc1-
```

```
0="urn:us:gov:dod:don:enterprise:code:temperatureMeasureUnitCode:1:0"
xmlns:cct="urn:us:gov:dod:don:enterprise:cct:1:0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:bie1-
0="urn:us:gov:dod:don:enterprise:bie:1:0" xmlns:qdt1-0="urn:us:gov:dod:don:enterprise:qdt:1:0"
xmlns:udt1-0="urn:us:gov:dod:don:enterprise:udt:1:0" xmlns:iso3166-1="urn:iso:codelist:3166-
1:CountryIdentificationCode:1:0" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0.0">
    <xsd:import namespace="urn:us:gov:dod:don:enterprise:qdt:1:0" schemaLocation="DON-
Enterprise-QDT-1.0.0.xsd"/>
    <xsd:import namespace="urn:iso:codelist:3166-1:CountryIdentificationCode:1:0"
schemaLocation="..\..\VCS\ISO3166.1-CodeList-CountryIdentificationCode-1.0.0.xsd"/>
    <xsd:import namespace="http://www.w3.org/1999/xlink" schemaLocation="..\..\VCS\W3C-Xlink-
hrefAttribute.xsd"/>
    <!-- ============ Global Attributes ================-->
    <xsd:attribute name="objectIDREF" type="xsd:token" use="required"/>
    <!--============ ABIEs ======================-->
    <!--*** ABIE Strike. Details ***-->
    <xsd:element name="Strike" type="bie1-0:StrikeType">
        <xsd:unique name="OrdnanceUnique">
            <xsd:selector xpath="bie1-0:AvailableOrdnance"/>
            <xsd:field xpath="bie1-0:OrdnanceID"/>
        </xsd:unique>
        <xsd:keyref name="TargetAssignedOrdnanceREF" refer="bie1-0:OrdnanceUnique">
            <xsd:selector xpath="bie1-0:Target/bie1-0:AssignedOrdnanceREF"/>
            <xsd:field xpath="@bie1-0:objectIDREF"/>
        </xsd:keyref>
    </xsd:element>
    <xsd:complexType name="StrikeType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:ABIEComponent>
                    <cdp:UID>A0001</cdp:UID>
                    <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Strike. Details</cdp:DictionaryEntryName>
                    <cdp:Definition>An aggregation of information describing a offensive action
carried against a designated target.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:ObjectClass>Strike</cdp:ObjectClass>
                </cdp:ABIEComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="bie1-0:JulianDay">
                <xsd:annotation>
                    <xsd:documentation>
                        <cdp:BBIEComponent>
                            <cdp:UID>B0001</cdp:UID>
                            <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                            <cdp:DictionaryEntryName>Strike. Julian_ Day.
Text</cdp:DictionaryEntryName>
                            <cdp:Definition>A 3-digit number representing the number of the
calendar day on which a Strike action will or did occur.</cdp:Definition>
                            <cdp:VersionData>
                                <cdp:VersionID>1.0</cdp:VersionID>
                                <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                            </cdp:VersionData>
                            <cdp:ObjectClass>Strike</cdp:ObjectClass>
                            <cdp:QualifierPropertyTerm>Julian</cdp:QualifierPropertyTerm>
                            <cdp:PropertyTerm>Day</cdp:PropertyTerm>
```

```
                                <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                                <cdp:Cardinality>1..1</cdp:Cardinality>
                                <cdp:Example>023</cdp:Example>
                            </cdp:BBIEComponent>
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element ref="bie1-0:Target" maxOccurs="unbounded">
                    <xsd:annotation>
                        <xsd:documentation>
                            <cdp:ASBIEComponent>
                                <cdp:UID>AS0001</cdp:UID>
                                <cdp:CategoryCode>ASBIE</cdp:CategoryCode>
                                <cdp:DictionaryEntryName>Strike. Target</cdp:DictionaryEntryName>
                                <cdp:Definition>The unit or location at which a strike is
directed.</cdp:Definition>
                                <cdp:VersionData>
                                    <cdp:VersionID>1.0</cdp:VersionID>
                                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                                </cdp:VersionData>
                                <cdp:ObjectClass>Strike</cdp:ObjectClass>
                                <cdp:AssociationName>Target</cdp:AssociationName>
                                <cdp:AssociatedObjectClass>Target</cdp:AssociatedObjectClass>
                                <cdp:Cardinality>1..n</cdp:Cardinality>
                            </cdp:ASBIEComponent>
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element ref="bie1-0:AvailableOrdnance" maxOccurs="unbounded">
                    <xsd:annotation>
                        <xsd:documentation>
                            <cdp:ASBIEComponent>
                                <cdp:UID>AS0002</cdp:UID>
                                <cdp:CategoryCode>ASBIE</cdp:CategoryCode>
                                <cdp:DictionaryEntryName>Strike. Available.
Ordnance</cdp:DictionaryEntryName>
                                <cdp:Definition>An aggregation of data describing an item of ordnance
that is available for use by a specific strike action.</cdp:Definition>
                                <cdp:VersionData>
                                    <cdp:VersionID>1.0</cdp:VersionID>
                                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                                </cdp:VersionData>
                                <cdp:ObjectClass>Strike</cdp:ObjectClass>
                                <cdp:AssociationName>Available</cdp:AssociationName>
                                <cdp:AssociatedObjectClass>Available_
Ordnance</cdp:AssociatedObjectClass>
                                <cdp:Cardinality>1..1</cdp:Cardinality>
                            </cdp:ASBIEComponent>
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:choice>
                    <xsd:element ref="bie1-0:AreaGeographicMap">
                        <xsd:annotation>
                            <xsd:documentation>
                                <cdp:ASBIEComponent>
                                    <cdp:UID>AS0003</cdp:UID>
                                    <cdp:CategoryCode>ASBIE</cdp:CategoryCode>
                                    <cdp:DictionaryEntryName>Target. Area. Geographic
Map</cdp:DictionaryEntryName>
                                    <cdp:Definition>A geographic map of an area surrounding an
intended target.</cdp:Definition>
```

```
                                    <cdp:VersionData>
                                        <cdp:VersionID>1.0</cdp:VersionID>
                                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                                    </cdp:VersionData>
                                    <cdp:ObjectClass>Target</cdp:ObjectClass>
                                    <cdp:AssociationName>Area</cdp:AssociationName>
                                    <cdp:AssociatedObjectClass>Geographic
Map</cdp:AssociatedObjectClass>
                                    <cdp:Cardinality>0..1</cdp:Cardinality>
                                </cdp:ASBIEComponent>
                            </xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element ref="bie1-0:AreaGeographicMapLink">
                        <xsd:annotation>
                            <xsd:documentation>
                                <cdp:AssociationComponent>
                                    <cdp:UID>AS0004</cdp:UID>
                                    <cdp:CategoryCode>LINK</cdp:CategoryCode>
                                    <cdp:DictionaryEntryName>Target. Area. Geographic Map
Link</cdp:DictionaryEntryName>
                                    <cdp:Definition>An xLink reference to a geographic map of the ares
surrounding and intended target.</cdp:Definition>
                                    <cdp:VersionData>
                                        <cdp:VersionID>1.0</cdp:VersionID>
                                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                                    </cdp:VersionData>
                                    <cdp:ObjectClass>Target</cdp:ObjectClass>
                                    <cdp:AssociationName>Area</cdp:AssociationName>
                                    <cdp:AssociatedObjectClass>Geographic Map
Link</cdp:AssociatedObjectClass>
                                    <cdp:Cardinality>0..1</cdp:Cardinality>
                                </cdp:AssociationComponent>
                            </xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:choice>
            </xsd:sequence>
        </xsd:complexType>
        <!--*** ABIE Target. Details ***-->
        <xsd:element name="Target" type="bie1-0:TargetType"/>
        <xsd:complexType name="TargetType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:ABIEComponent>
                        <cdp:UID>A0002</cdp:UID>
                        <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Target. Details</cdp:DictionaryEntryName>
                        <cdp:Definition>An unit or location designated for attack.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>string</cdp:ObjectClass>
                    </cdp:ABIEComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                <xsd:element name="TargetID" type="bie1-0:TargetIDType"/>
                <xsd:element ref="bie1-0:Description">
                    <xsd:annotation>
                        <xsd:documentation>
```

```
                        <cdp:BBIEComponent>
                            <cdp:UID>B0002</cdp:UID>
                            <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                            <cdp:DictionaryEntryName>Target. Description.
Text</cdp:DictionaryEntryName>
                            <cdp:Definition>A free-text description of a target.</cdp:Definition>
                            <cdp:VersionData>
                                <cdp:VersionID>1.0</cdp:VersionID>
                                <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                            </cdp:VersionData>
                            <cdp:ObjectClass>Target</cdp:ObjectClass>
                            <cdp:PropertyTerm>Description</cdp:PropertyTerm>
                            <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                            <cdp:Cardinality>1..1</cdp:Cardinality>
                        </cdp:BBIEComponent>
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element ref="bie1-0:AssignedOrdnanceREF">
                <xsd:annotation>
                    <xsd:documentation>
                        <cdp:ASBIEComponent>
                            <cdp:UID>AS0005</cdp:UID>
                            <cdp:CategoryCode>ASBIE</cdp:CategoryCode>
                            <cdp:DictionaryEntryName>Target. Assigned. Ordnance
Reference</cdp:DictionaryEntryName>
                            <cdp:Definition>A pointer to an ordnance item that has been assigned to
a target.</cdp:Definition>
                            <cdp:VersionData>
                                <cdp:VersionID>1.0</cdp:VersionID>
                                <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                            </cdp:VersionData>
                            <cdp:ObjectClass>Target</cdp:ObjectClass>
                            <cdp:AssociationName>Assigned</cdp:AssociationName>
                            <cdp:AssociatedObjectClass>Ordnance
Reference</cdp:AssociatedObjectClass>
                            <cdp:Cardinality>1..1</cdp:Cardinality>
                        </cdp:ASBIEComponent>
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element ref="bie1-0:CountryCode">
                <xsd:annotation>
                    <xsd:documentation>
                        <cdp:BBIEComponent>
                            <cdp:UID>B0003</cdp:UID>
                            <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                            <cdp:DictionaryEntryName>Target. Country.
Code</cdp:DictionaryEntryName>
                            <cdp:Definition>A code representing the country to which or within which
a target belongs.</cdp:Definition>
                            <cdp:VersionData>
                                <cdp:VersionID>1.0</cdp:VersionID>
                                <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                            </cdp:VersionData>
                            <cdp:ObjectClass>Target</cdp:ObjectClass>
                            <cdp:PropertyTerm>Country</cdp:PropertyTerm>
                            <cdp:RepresentationTerm>Code</cdp:RepresentationTerm>
                            <cdp:Cardinality>1..1</cdp:Cardinality>
                        </cdp:BBIEComponent>
                    </xsd:documentation>
                </xsd:annotation>
```

```
                </xsd:element>
                <xsd:element ref="bie1-0:SurfaceTemperatureMeasure" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>
                            <cdp:BBIEComponent>
                                <cdp:UID>B0004</cdp:UID>
                                <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                                <cdp:DictionaryEntryName>Target. Surface_ Temperature.
Measure</cdp:DictionaryEntryName>
                                <cdp:Definition>The earth surface temperature in at a target
location.</cdp:Definition>
                                <cdp:VersionData>
                                    <cdp:VersionID>1.0</cdp:VersionID>
                                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                                </cdp:VersionData>
                                <cdp:ObjectClass>Target</cdp:ObjectClass>
                                <cdp:QualifierPropertyTerm>Surface</cdp:QualifierPropertyTerm>
                                <cdp:PropertyTerm>Temperature</cdp:PropertyTerm>
                                <cdp:RepresentationTerm>Measure</cdp:RepresentationTerm>
                                <cdp:Cardinality>0..1</cdp:Cardinality>
                            </cdp:BBIEComponent>
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <!-- TargetID BBIE not documented here since element name has object class already.
-->
            </xsd:sequence>
        </xsd:complexType>
        <!--    BBIE Target. Identifier-->
        <xsd:complexType name="TargetIDType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:BBIEComponent>
                        <cdp:UID>B0005</cdp:UID>
                        <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Target. Identifier</cdp:DictionaryEntryName>
                        <cdp:Definition>A normalized character string that uniquely identifies a
target.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Target</cdp:ObjectClass>
                        <cdp:PropertyTerm>Identifier</cdp:PropertyTerm>
                        <cdp:RepresentationTerm>Identifier</cdp:RepresentationTerm>
                        <cdp:Cardinality>1..1</cdp:Cardinality>
                    </cdp:BBIEComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="qdt1-0:HostileTargetIDType"/>
            </xsd:simpleContent>
        </xsd:complexType>
        <!--*** ABIE Ordnance. Details ***-->
        <xsd:element name="Ordnance" type="bie1-0:OrdnanceType"/>
        <xsd:complexType name="OrdnanceType">
            <xsd:sequence>
                <xsd:element ref="bie1-0:OrdnanceID">
                    <xsd:annotation>
                        <xsd:documentation>
                            <cdp:ABIEComponent>
                                <cdp:UID>A0003</cdp:UID>
```

```
                        <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Ordnance.
Details</cdp:DictionaryEntryName>
                        <cdp:Definition>An explosive device capable of being delivered to a
target.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Ordnance</cdp:ObjectClass>
                    </cdp:ABIEComponent>
                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element ref="bie1-0:Nomenclature">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:BBIEComponent>
                        <cdp:UID>B0006</cdp:UID>
                        <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Ordnance. Nomenclature.
Text</cdp:DictionaryEntryName>
                        <cdp:Definition>The official assigned nomenclature of an Ordnance
system.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Ordnance</cdp:ObjectClass>
                        <cdp:PropertyTerm>Nomenclature</cdp:PropertyTerm>
                        <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                        <cdp:Cardinality>1..1</cdp:Cardinality>
                    </cdp:BBIEComponent>
                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<!--Qualifed ABIE Available_ Ordnance. Detals ***-->
<xsd:element name="AvailableOrdnance" type="bie1-0:AvailableOrdnanceType"/>
<xsd:complexType name="AvailableOrdnanceType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:ABIEComponent>
                <cdp:UID>A0004</cdp:UID>
                <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                <cdp:DictionaryEntryName>Available_ Ordnance.
Details</cdp:DictionaryEntryName>
                <cdp:Definition>Ordnance that is available for some purpose.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:QualifierObjectClass>Available</cdp:QualifierObjectClass>
                <cdp:ObjectClass>Ordnance</cdp:ObjectClass>
            </cdp:ABIEComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="bie1-0:OrdnanceType"/>
    </xsd:complexContent>
</xsd:complexType>
```

```
    <!-- Reference ABIE: Assigned Ordnance_ Reference. Details  ***-->
    <xsd:element name="AssignedOrdnanceREF" type="bie1-0:AssignedOrdnanceREFType"/>
    <xsd:complexType name="AssignedOrdnanceREFType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:ABIEComponent>
                    <cdp:UID>A0005</cdp:UID>
                    <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Assigned Ordnance_ Reference.
Details</cdp:DictionaryEntryName>
                    <cdp:Definition>string</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>string</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:QualifierObjectClass/>
                    <cdp:ObjectClass>string</cdp:ObjectClass>
                    <cdp:BusinessTerm>string</cdp:BusinessTerm>
                    <cdp:UsageRule>string</cdp:UsageRule>
                </cdp:ABIEComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:attribute ref="bie1-0:objectIDREF" use="required"/>
    </xsd:complexType>
    <!--BBIE Ordnance. Identifier-->
    <xsd:element name="OrdnanceID" type="bie1-0:OrdnanceIDType"/>
    <xsd:complexType name="OrdnanceIDType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:BBIEComponent>
                    <cdp:UID>B0007</cdp:UID>
                    <cdp:CategoryCode>BBIE</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Ordnance. Identifier</cdp:DictionaryEntryName>
                    <cdp:Definition>A normalized string that uniquely identifies an item of
ordnance.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:ObjectClass>Ordnance</cdp:ObjectClass>
                    <cdp:PropertyTerm>Identifier</cdp:PropertyTerm>
                    <cdp:RepresentationTerm>Identifier</cdp:RepresentationTerm>
                    <cdp:Cardinality>1..1</cdp:Cardinality>
                </cdp:BBIEComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="qdt1-0:OrdnanceIDType"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- *** ABIE Geographic Map ***  -->
    <xsd:element name="GeographicMap" type="bie1-0:GeographicMapType"/>
    <xsd:complexType name="GeographicMapType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:ABIEComponent>
                    <cdp:UID>A0006</cdp:UID>
                    <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Geographic Map.
Details</cdp:DictionaryEntryName>
                    <cdp:Definition>A Scalable Vector Graphic formated representation of a
geographic map.</cdp:Definition>
```

```
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Geographic Map</cdp:ObjectClass>
                    </cdp:ABIEComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                <xsd:any namespace="http://www.w3.org/2000/svg"/>
            </xsd:sequence>
        </xsd:complexType>
        <!--*** Qualiified ABIE: Area_ Geographic Map. Details-->
        <xsd:element name="AreaGeographicMap"/>
        <xsd:complexType name="AreaGeographicMapType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:ABIEComponent>
                        <cdp:UID>A0007</cdp:UID>
                        <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Area_ Geographic Map.
Details</cdp:DictionaryEntryName>
                        <cdp:Definition>A geographic map of an area or region.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:QualifierObjectClass>Area</cdp:QualifierObjectClass>
                        <cdp:ObjectClass>Geographic Map</cdp:ObjectClass>
                    </cdp:ABIEComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:complexContent>
                <xsd:extension base="bie1-0:GeographicMapType"/>
            </xsd:complexContent>
        </xsd:complexType>
        <!--*** ABIE:  Area Geographic Map_ Link. Details-->
        <xsd:element name="AreaGeographicMapLink" type="bie1-0:AreaGeographicMapLinkType"/>
        <xsd:complexType name="AreaGeographicMapLinkType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:ABIEComponent>
                        <cdp:UID>A0008</cdp:UID>
                        <cdp:CategoryCode>ABIE</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Area Geographic Map_ Link.
Details</cdp:DictionaryEntryName>
                        <cdp:Definition>A xlink reference to a geographic map of an area or
region.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:QualifierObjectClass>Area Geographic Map</cdp:QualifierObjectClass>
                        <cdp:ObjectClass>Link</cdp:ObjectClass>
                    </cdp:ABIEComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                <xsd:any namespace="http://www.w3.org/1999/xlink"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="http://www.w3.org/1999/xlink"/>
        </xsd:complexType>
```

```
    <!--======= BBIE Property Global Elements and ComplexTypes  ==============-->
    <xsd:element name="Description" type="bie1-0:DescriptionType"/>
    <xsd:complexType name="DescriptionType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:BBIEPropertyComponent>
                    <cdp:UID>B0008</cdp:UID>
                    <cdp:CategoryCode>BBIEProperty</cdp:CategoryCode>
                    <cdp:Definition>A string providing descriptive information about some
entity.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:PropertyTerm>Description</cdp:PropertyTerm>
                    <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                </cdp:BBIEPropertyComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="udt1-0:TextType"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:element name="Nomenclature" type="bie1-0:NomenclatureType"/>
    <xsd:complexType name="NomenclatureType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:BBIEPropertyComponent>
                    <cdp:UID>B0009</cdp:UID>
                    <cdp:CategoryCode>BBIEProperty</cdp:CategoryCode>
                    <cdp:Definition>An official, assigned nomenclature by which a DOD materiel
items is referred.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:PropertyTerm>Nomenclature</cdp:PropertyTerm>
                    <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                </cdp:BBIEPropertyComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="udt1-0:TextType"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:element name="CountryCode" type="bie1-0:CountryCodeType"/>
    <xsd:complexType name="CountryCodeType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:BBIEPropertyComponent>
                    <cdp:UID>B0010</cdp:UID>
                    <cdp:CategoryCode>BBIEProperty</cdp:CategoryCode>
                    <cdp:Definition>A code representing a country.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:PropertyTerm>Country</cdp:PropertyTerm>
                    <cdp:RepresentationTerm>Code</cdp:RepresentationTerm>
                </cdp:BBIEPropertyComponent>
            </xsd:documentation>
        </xsd:annotation>
```

```
            <xsd:choice>
                <xsd:element ref="iso3166-1:ISO31661CountryCode"/>
            </xsd:choice>
        </xsd:complexType>
        <xsd:element name="JulianDay" type="bie1-0:JulianDayType"/>
        <xsd:complexType name="JulianDayType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:BBIEPropertyComponent>
                        <cdp:UID>B0011</cdp:UID>
                        <cdp:CategoryCode>BBIEProperty</cdp:CategoryCode>
                        <cdp:Definition>A string representing the number of days that have elapsed in
the current Julian calendar year.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:QualifierPropertyTerm>Julian</cdp:QualifierPropertyTerm>
                        <cdp:PropertyTerm>Day</cdp:PropertyTerm>
                        <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                    </cdp:BBIEPropertyComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="qdt1-0:JulianDayText"/>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:element name="SurfaceTemperatureMeasure" type="qdt1-
0:SurfaceTemperatureMeasureType"/>
</xsd:schema>
```

1.1.6 DON-Enterprise-Qualified-DataTypes-1.0.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Department of the Navy (DON) XML Naming and Design Rules (NDRs)
  DocType - DON Enterprise Qualified Data Types (QDT)
  Purpose - defines XSD complexTypes representing CCTS qualified data types that have been
harmonized into the DON Enterprise Namespace.

  Version - 1.0.0
  Version Description - Initial release of Schema Module
  Date - 3/19/2004

  This XML schema is provided as an example of the DON Enterprise QDT Schema. It is an
example only. The actual Schema will be published and maintained by the  BSC upon approval of
the DON NDRs.
 -->
<xsd:schema targetNamespace="urn:us:gov:dod:don:enterprise:qdt:1:0" xmlns:udt1-
0="urn:us:gov:dod:don:enterprise:udt:1:0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:qdt1-0="urn:us:gov:dod:don:enterprise:qdt:1:0" xmlns:tmuc1-
0="urn:us:gov:dod:don:enterprise:code:temperatureMeasureUnitCode:1:0"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0.0">
    <xsd:import namespace="urn:us:gov:dod:don:enterprise:udt:1:0" schemaLocation="DON-
Enterprise-UDT-1.0.0.xsd"/>
    <xsd:import
namespace="urn:us:gov:dod:don:enterprise:code:temperatureMeasureUnitCode:1:0"
schemaLocation="DON-Enterprise-CodeList-TemperatureMeasureUnitCode-1.0.0.xsd"/>
    <!-- ============ Data Types ================== -->
    <!--***  Ordnance_ Identifier. Type  ***-->
    <xsd:complexType name="OrdnanceIDType">
```

```
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:IDDataTypeComponent>
                        <cdp:UID>QDT0001</cdp:UID>
                        <cdp:CategoryCode>QDT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Ordnance_ Identifier.
Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A normalized character string that uniquely identifies an item of
Ordnance</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>string</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:QualifierDataType>Ordnance</cdp:QualifierDataType>
                        <cdp:RepresentationTerm>Identifier</cdp:RepresentationTerm>
                        <cdp:IDSchemeRestriction>
                            <cdp:AgencyID>EX-001</cdp:AgencyID>
                            <cdp:AgencyName>Some DON Organization</cdp:AgencyName>

    <cdp:IDSchemeDataURI>http://www.someURI.mil</cdp:IDSchemeDataURI>
                            <cdp:URI>http://www.someURI.mil</cdp:URI>
                            <cdp:VersionID>1.0</cdp:VersionID>
                        </cdp:IDSchemeRestriction>
                    </cdp:IDDataTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction base="udt1-0:IdentifierType">
                    <xsd:pattern value="[A-Z]\d{5}"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <!--*** Hostile Target_ Identifier. Type ***-->
        <xsd:complexType name="HostileTargetIDType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:DataTypeComponent>

                    </cdp:DataTypeComponent>
                    <cdp:IDDataTypeComponent>
                        <cdp:UID>QDT0002</cdp:UID>
                        <cdp:CategoryCode>QDT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Hostile Target_ Identifier.
Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A normalized character string that uniquely identifies a target
designated as hostile.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:QualifierDataType>Hostile Target</cdp:QualifierDataType>
                        <cdp:RepresentationTerm>Identifier</cdp:RepresentationTerm>
                        <cdp:IDSchemeRestriction>
                            <cdp:AgencyID>EX-001</cdp:AgencyID>
                            <cdp:AgencyName>Some DON Organization</cdp:AgencyName>

    <cdp:IDSchemeDataURI>http://www.someURI.mil</cdp:IDSchemeDataURI>
                            <cdp:URI>http://www.someURI.mil</cdp:URI>
                            <cdp:VersionID>1.0</cdp:VersionID>
                        </cdp:IDSchemeRestriction>
                    </cdp:IDDataTypeComponent>
                </xsd:documentation>
```

```
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction base="udt1-0:IdentifierType">
                    <xsd:pattern value="[A-Z][0-9][A-Z]\d{3}"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <!--*** Julian_ Date. Type ***-->
        <xsd:complexType name="JulianDateText">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:DataTypeComponent>
                        <cdp:UID>QDT0003</cdp:UID>
                        <cdp:CategoryCode>QDT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Julian Date_ Text.
Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A 4-position string where the first position indicates the last digit
of the current year, and the next three digits represent a day in the 365 day Julian
calendar</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:QualifierDataType>Julian Date</cdp:QualifierDataType>
                        <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                        <cdp:Example>4010</cdp:Example>
                        <cdp:RestrictionList>
                            <cdp:ContentComponentRestriction>
                                <cdp:RestrictionType>PATTERN</cdp:RestrictionType>
                                <cdp:RestrictionDescription>A 4 position number where the first position
represents is a number 0-9, and the last three are between the values of 000 and
365.</cdp:RestrictionDescription>
                            </cdp:ContentComponentRestriction>
                        </cdp:RestrictionList>
                    </cdp:DataTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction base="udt1-0:TextType">
                    <xsd:pattern value="[0-9]+0[0-9][0-9]|1[0-9][0-9]|2[0-9][0-9]|3[0-5][0-9]|36[0-5]"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType name="JulianDayText">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:DataTypeComponent>
                        <cdp:UID>QDT0004</cdp:UID>
                        <cdp:CategoryCode>QDT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Julian Day_ Text. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A 3-position string that represents a day in the 365 day Julian
calendar</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:QualifierDataType>Julian Day</cdp:QualifierDataType>
                        <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                        <cdp:Example>010</cdp:Example>
                        <cdp:RestrictionList>
                            <cdp:ContentComponentRestriction>
                                <cdp:RestrictionType>PATTERN</cdp:RestrictionType>
```

```
                        <cdp:RestrictionDescription>A 3 position number where the between the
values of 000 and 365.</cdp:RestrictionDescription>
                    </cdp:ContentComponentRestriction>
                </cdp:RestrictionList>
            </cdp:DataTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:restriction base="qdt1-0:JulianDateText">
            <xsd:pattern value="0[0-9][0-9]|1[0-9][0-9]|2[0-9][0-9]|3[0-5][0-9]|36[0-5]"/>
        </xsd:restriction>
    </xsd:simpleContent>
</xsd:complexType>
<!--*** Temperature_ Measure. Type ***-->
<xsd:complexType name="TemperatureMeasureType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:DataTypeComponent>
                <cdp:UID>QDT0005</cdp:UID>
                <cdp:CategoryCode>QDT</cdp:CategoryCode>
                <cdp:DictionaryEntryName>Temperature_ Measure.
Type</cdp:DictionaryEntryName>
                <cdp:Definition>A measurement of an object's temperature to the nearest 100th
of a degree.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:QualifierDataType>Temperature</cdp:QualifierDataType>
                <cdp:RepresentationTerm>Measure</cdp:RepresentationTerm>
                <cdp:Example>32.00</cdp:Example>
            </cdp:DataTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:restriction base="udt1-0:MeasureType">
            <xsd:fractionDigits value="2"/>
            <xsd:totalDigits value="5"/>
            <xsd:attribute name="unitCode" type="tmuc1-0:TemperatureMeasureUnitCode"/>
        </xsd:restriction>
    </xsd:simpleContent>
</xsd:complexType>
<!--*** SurfaceTemperature_ Measure. Type ***-->
<xsd:complexType name="SurfaceTemperatureMeasureType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:DataTypeComponent>
                <cdp:UID>QDT0006</cdp:UID>
                <cdp:CategoryCode>QDT</cdp:CategoryCode>
                <cdp:DictionaryEntryName>SurfaceTemperature_ Measure.
Type</cdp:DictionaryEntryName>
                <cdp:Definition>The temperature as read at the surface of some
boundary.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:QualifierDataType>SurfaceTemperature</cdp:QualifierDataType>
                <cdp:RepresentationTerm>Measure</cdp:RepresentationTerm>
            </cdp:DataTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
```

```
            <xsd:simpleContent>
                <xsd:extension base="qdt1-0:TemperatureMeasureType"/>
            </xsd:simpleContent>
        </xsd:complexType>
</xsd:schema>
```

1.1.7 DON-Enterprise-Unqualified-DataTypes-1.0.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Department of the Navy (DON) XML Naming and Design Rules (NDRs)
  DocType - DON Enterprise Unqualified Data Types (UDT)
  Purpose - defines XML Schema complex types for the set of DON BSC approved
  Unqualified Data Types. These map directly to the primary and secondary representation terms
defined in the Core Components Technical Specification (CCTS)

  Version - 1.0
  Version Description - Initial release of Schema Module
  Date - 3/19/2004

  This XML schema was developed based on the UBL Representation Terms beta release
  Schema as  documented in the following copyright statement. It represents the
  normative set of allowable Unqualified Data Types for the DON NDRs version 2.0. These
  complex types are based on the set of primary and secondary representatino terms
  defined in table 8-1 of the CCTS Version 2.0.1
 -->
<xsd:schema targetNamespace="urn:us:gov:dod:don:enterprise:udt:1:0"
xmlns="urn:us:gov:dod:don:enterprise:udt:1:0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0.0">
    <!-- ===== UDT: AmountType ===== -->
    <xsd:element name="Amount" type="AmountType"/>
    <xsd:complexType name="AmountType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT001</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Amount. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A number of monetary units specified in a currency where the
unit of the currency is explicit or implied.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Amount</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:decimal">
                <xsd:attribute name="currencyID" type="xsd:token" use="optional"/>
                <xsd:attribute name="codeListVersionID" type="xsd:token" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: BinaryObjectType ===== -->
    <xsd:element name="BinaryObject" type="BinaryObjectType"/>
    <xsd:complexType name="BinaryObjectType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
```

```
                    <cdp:UID>UDT002</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Binary Object. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A set of finite-length sequences of binary
octets.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Binary Object</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:base64Binary">
                <xsd:attribute name="format" type="xsd:token" use="optional"/>
                <xsd:attribute name="mimeCode" type="xsd:token" use="optional"/>
                <xsd:attribute name="encodingCode" type="xsd:token" use="optional"/>
                <xsd:attribute name="characterSetCode" type="xsd:token" use="optional"/>
                <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
                <xsd:attribute name="filename" type="xsd:token" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: GraphicType ===== -->
    <xsd:element name="Graphic" type="GraphicType"/>
    <xsd:complexType name="GraphicType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT003</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Graphic. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A diagram, graph, mathematical curves, or similar
representation.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Graphic</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction base="BinaryObjectType">
                <xsd:attribute name="characterSetCode" use="prohibited"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: PictureType ===== -->
    <xsd:element name="Picture" type="PictureType"/>
    <xsd:complexType name="PictureType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT004</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Picture. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A visual representation of a person, object, or
scene.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
```

```
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Picture</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction base="BinaryObjectType">
                <xsd:attribute name="characterSetCode" use="prohibited"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: SoundType ===== -->
    <xsd:element name="SoundBinaryObject" type="SoundType"/>
    <xsd:complexType name="SoundType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT005</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Sound. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>An audio encoding represented as a binary
object.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Sound</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction base="BinaryObjectType">
                <xsd:attribute name="characterSetCode" use="prohibited"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: VideoType ===== -->
    <xsd:element name="Video" type="VideoType"/>
    <xsd:complexType name="VideoType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT006</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Video. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>Relating to the recording, reproducing or broadcasting of visual
images on magnetic tape or digitially.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Video</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction base="BinaryObjectType">
                <xsd:attribute name="characterSetCode" use="prohibited"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
```

```
<!-- ===== UDT: CodeType ===== -->
<xsd:element name="CodeContent" type="CodeContentType"/>
<xsd:simpleType name="CodeContentType">
    <xsd:restriction base="xsd:token"/>
</xsd:simpleType>
<xsd:complexType name="CodeType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:DataTypeComponent>
                <cdp:UID>UDT007 </cdp:UID>
                <cdp:CategoryCode>UDT</cdp:CategoryCode>
                <cdp:DictionaryEntryName>Code. Type</cdp:DictionaryEntryName>
                <cdp:Definition>A character string (letters, figures, or symbols) that for brevity
and/or languange independence may be used to represent or replace a definitive value or text of an
attribute together with relevant supplementary information.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:RepresentationTerm>Code</cdp:RepresentationTerm>
            </cdp:DataTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="CodeContentType"/>
    </xsd:simpleContent>
</xsd:complexType>
<!-- ===== UDT: DateTimeType ===== -->
<xsd:element name="DateTime" type="DateTimeType"/>
<xsd:simpleType name="DateTimeType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:CoreComponentTypeComponent>
                <cdp:UID>UDT008</cdp:UID>
                <cdp:CategoryCode>UDT</cdp:CategoryCode>
                <cdp:DictionaryEntryName>Date Time. Type</cdp:DictionaryEntryName>
                <cdp:Definition>A particular point in the progression of time together with the
relevant supplementary information.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:ObjectClass>Date Time</cdp:ObjectClass>
            </cdp:CoreComponentTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>
<!-- ===== UDT: DateType ===== -->
<xsd:element name="Date" type="DateType"/>
<xsd:simpleType name="DateType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:DataTypeComponent>
                <cdp:UID>UDT009</cdp:UID>
                <cdp:CategoryCode>UDT</cdp:CategoryCode>
                <cdp:DictionaryEntryName>Date. Type</cdp:DictionaryEntryName>
                <cdp:Definition>A representation of the progression of time according the
Gregorian calendar.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
```

```
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Date</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:date"/>
    </xsd:simpleType>
    <!-- ===== UDT: TimeType ===== -->
    <xsd:element name="Time" type="TimeType"/>
    <xsd:simpleType name="TimeType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0010</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Time. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A measurement of the interval between which events occurs
within a specific calendar day.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Time</cdp:RepresentationTerm>
                    <cdp:example>6:00 am</cdp:example>
                    <cdp:example>06:00:00</cdp:example>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:time"/>
    </xsd:simpleType>
    <!-- ===== UDT: IdentifierType ===== -->
    <xsd:element name="Identifier" type="IdentifierType"/>
    <xsd:complexType name="IdentifierType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0011</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Identifier. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A character string to identify and distinguish uniquely, one
instance of an object in an identification scheme from all other objects in the same scheme together
with relevant supplementary information.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Identifier</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:normalizedString"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: IndicatorType ===== -->
    <xsd:element name="Indicator" type="IndicatorType"/>
    <xsd:simpleType name="IndicatorType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:CoreComponentTypeComponent>
                    <cdp:UID>UDT012</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
```

```
                        <cdp:DictionaryEntryName>Indicator. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A list of two mutually exclusive Boolean values that express the
only possible states of a Property.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Indicator</cdp:ObjectClass>
                    </cdp:CoreComponentTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:boolean"/>
        </xsd:simpleType>
        <!-- ===== UDT: MeasureType ===== -->
        <xsd:element name="Measure" type="MeasureType"/>
        <xsd:complexType name="MeasureType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:DataTypeComponent>
                        <cdp:UID>UDT0013</cdp:UID>
                        <cdp:CategoryCode>UDT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Measure. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A numeric value determined by measuring an object along with
the specified unit of measure.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:RepresentationTerm>Measure</cdp:RepresentationTerm>
                    </cdp:DataTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:decimal">
                    <xsd:attribute name="unitCode" type="xsd:token" use="optional"/>
                    <xsd:attribute name="unitCodeListVersionID" type="xsd:token" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!-- ===== UDT: NumericType ===== -->
        <xsd:element name="Numeric" type="NumericType"/>
        <xsd:complexType name="NumericType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:DataTypeComponent>
                        <cdp:UID>UDT0014</cdp:UID>
                        <cdp:CategoryCode>UDT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Numeric. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>Numeric information that is assigned or is determined by
calculation, counting, or sequencing. It does not require a unit of quantity or unit of
measure.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:RepresentationTerm>Numeric</cdp:RepresentationTerm>
                    </cdp:DataTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:decimal">
                    <xsd:attribute name="numericFormat" type="xsd:string" use="optional"/>
```

```
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: ValueType ===== -->
    <xsd:element name="Value" type="ValueType"/>
    <xsd:complexType name="ValueType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0015</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Value. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>Assigned or calculated numeric information representing a
quantifiable characteristic of an object.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Value</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="NumericType"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: PercentType ===== -->
    <xsd:element name="Percent" type="PercentType"/>
    <xsd:complexType name="PercentType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0016</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Percentage. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>Numeric information representing a fraction or ratio with an
implied denominator of 100</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Percentage</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="NumericType"/>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: RateType ===== -->
    <xsd:element name="Rate" type="RateType"/>
    <xsd:complexType name="RateType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0017</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Rate. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>Numeric information that represents the the ratio between an
object's time-dependant characteristic expressed in base units per specified unit
time.</cdp:Definition>
                    <cdp:VersionData>
```

```
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Percentage</cdp:RepresentationTerm>
                    <cdp:example>60 with rate unit = miles, base unit = hour.</cdp:example>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="NumericType">
                <xsd:attribute name="rateUnitCode" type="xsd:normalizedString" use="required"/>
                <xsd:attribute name="rateBasisUnitCode" type="xsd:normalizedString"
use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: QuantityType ===== -->
    <xsd:element name="Quantity" type="QuantityType"/>
    <xsd:complexType name="QuantityType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0018</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Quantity. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A counted number of non-monetary units possibly including
fractions.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Quantity</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:decimal">
                <xsd:attribute name="unitCode" type="xsd:token" use="optional"/>
                <xsd:attribute name="unitCodeListID" type="xsd:token" use="optional"/>
                <xsd:attribute name="unitCodeListAgencyID" type="xsd:token" use="optional"/>
                <xsd:attribute name="unitCodeListAgencyName" type="xsd:token" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== UDT: TextType ===== -->
    <xsd:element name="Text" type="TextType"/>
    <xsd:complexType name="TextType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:DataTypeComponent>
                    <cdp:UID>UDT0019</cdp:UID>
                    <cdp:CategoryCode>UDT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Text. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A character string (i.e. a finite set of characters) generally in the
form of words of a language.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:RepresentationTerm>Text</cdp:RepresentationTerm>
                </cdp:DataTypeComponent>
            </xsd:documentation>
```

```
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="languageID" type="xsd:language" use="optional"/>
                    <xsd:attribute name="languageLocaleID" type="xsd:token" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!-- ===== UDT: NameType ===== -->
        <xsd:element name="Name" type="NameType"/>
        <xsd:complexType name="NameType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:DataTypeComponent>
                        <cdp:UID>UDT0020</cdp:UID>
                        <cdp:CategoryCode>UDT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Name. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A word or words by which an entity is designated and
distinguished from others.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>ki
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:RepresentationTerm>Name</cdp:RepresentationTerm>
                    </cdp:DataTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction base="TextType">
                    <xsd:attribute name="languageID" type="xsd:language" use="optional"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
</xsd:schema>
```

1.1.8 DON-Enterprise-CCT-1.0.0.xsd (For normalized reference **only**. This module will **not** be imported by the UDT Schema module.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Department of the Navy (DON) XML Naming and Design Rules (NDRs)
  DocType - DON Enterprise Core Component Types (CCT)
  Purpose - defines XML Schema complex types for the CCTS Core Component Types

  Version - 1.0
  Version Description - Initial release of Schema Module
  Date - 3/19/2004

  This XML schema was developed based on the UBL Core Component Types beta release
  Schema as  documented in the following copyright statement. It represents the
  normative Core Component Types Schema for the DON NDRs version 2.0.
 -->
<xsd:schema targetNamespace="urn:us:gov:dod:don:enterprise:cct:1:0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:cct="urn:us:gov:dod:don:enterprise:cct:1:0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1:0">
    <!-- ===== CCT: AmountType ===== -->
    <xsd:element name="Amount" type="cct:AmountType"/>
    <xsd:complexType name="AmountType">
```

```
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:CoreComponentTypeComponent>
                        <cdp:UID>CCT001</cdp:UID>
                        <cdp:CategoryCode>CCT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Amount. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A number of monetary units specified in a currency where the
unit of currency is explicit or implied</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Amount</cdp:ObjectClass>
                    </cdp:CoreComponentTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:decimal">
                    <xsd:attribute name="currencyID" type="xsd:token" use="optional"/>
                    <xsd:attribute name="codeListVersionID" type="xsd:token" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!-- ===== CCT: BinaryObjectType ===== -->
        <xsd:element name="BinaryObject" type="cct:BinaryObjectType"/>
        <xsd:complexType name="BinaryObjectType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:CoreComponentTypeComponent>
                        <cdp:UID>CCT002</cdp:UID>
                        <cdp:CategoryCode>CCT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Binary Object. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A set of finite-length sequences of binary
octets.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Binary Object</cdp:ObjectClass>
                    </cdp:CoreComponentTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:base64Binary">
                    <xsd:attribute name="format" type="xsd:token" use="optional"/>
                    <xsd:attribute name="mimeCode" type="xsd:token" use="optional"/>
                    <xsd:attribute name="encodingCode" type="xsd:token" use="optional"/>
                    <xsd:attribute name="characterSetCode" type="xsd:token" use="optional"/>
                    <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
                    <xsd:attribute name="filename" type="xsd:token" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!-- ===== CCT: CodeType ===== -->
        <xsd:element name="Code" type="cct:CodeType"/>
        <xsd:complexType name="CodeType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:CoreComponentTypeComponent>
                        <cdp:UID>CCT003</cdp:UID>
                        <cdp:CategoryCode>CCT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Code. Type</cdp:DictionaryEntryName>
```

```
                <cdp:Definition>A character string (letters, figures, or symbols) that for brevity
and/or languange independence may be used to represent or replace a definitive value or text of an
attribute together with relevant supplementary information.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:ObjectClass>Code</cdp:ObjectClass>
            </cdp:CoreComponentTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:normalizedString"/>
    </xsd:simpleContent>
</xsd:complexType>
<!-- ===== CCT: DateTimeType ===== -->
<xsd:element name="DateTime" type="cct:DateTimeType"/>
<xsd:simpleType name="DateTimeType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:CoreComponentTypeComponent>
                <cdp:UID>CCT004</cdp:UID>
                <cdp:CategoryCode>CCT</cdp:CategoryCode>
                <cdp:DictionaryEntryName>Date Time. Type</cdp:DictionaryEntryName>
                <cdp:Definition>A particular point in the progression of time together with the
relevant supplementary information.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:ObjectClass>Date Time</cdp:ObjectClass>
            </cdp:CoreComponentTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>
<!-- ===== CCT: IdentifierType ===== -->
<xsd:element name="Identifier" type="cct:IdentifierType"/>
<xsd:complexType name="IdentifierType">
    <xsd:annotation>
        <xsd:documentation>
            <cdp:CoreComponentTypeComponent>
                <cdp:UID>CCT005</cdp:UID>
                <cdp:CategoryCode>CCT</cdp:CategoryCode>
                <cdp:DictionaryEntryName>Identifier. Type</cdp:DictionaryEntryName>
                <cdp:Definition>A character string to identify and distinguish uniquely, one
instance of an object in an identification scheme from all other objects in the same scheme together
with relevant supplementary information.</cdp:Definition>
                <cdp:VersionData>
                    <cdp:VersionID>1.0</cdp:VersionID>
                    <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                </cdp:VersionData>
                <cdp:ObjectClass>Identifier</cdp:ObjectClass>
            </cdp:CoreComponentTypeComponent>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:normalizedString"/>
    </xsd:simpleContent>
</xsd:complexType>
<!-- ===== CCT: IndicatorType ===== -->
<xsd:element name="Indicator" type="cct:IndicatorType"/>
```

```
    <xsd:simpleType name="IndicatorType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:CoreComponentTypeComponent>
                    <cdp:UID>CCT006</cdp:UID>
                    <cdp:CategoryCode>CCT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Indicator. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A list of two mutually exclusive Boolean values that express the
only possible states of a Property.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:ObjectClass>Indicator</cdp:ObjectClass>
                </cdp:CoreComponentTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:boolean"/>
    </xsd:simpleType>
    <!-- ===== CCT: MeasureType ===== -->
    <xsd:element name="Measure" type="cct:MeasureType"/>
    <xsd:complexType name="MeasureType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:CoreComponentTypeComponent>
                    <cdp:UID>CCT007</cdp:UID>
                    <cdp:CategoryCode>CCT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Measure. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>A numeric value determined by measuring an object along with
the specified unit of measure.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:ObjectClass>Measure</cdp:ObjectClass>
                </cdp:CoreComponentTypeComponent>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:decimal">
                <xsd:attribute name="unitCode" type="xsd:token" use="optional"/>
                <xsd:attribute name="unitCodeListVersionID" type="xsd:token" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- ===== CCT: NumericType ===== -->
    <xsd:element name="NumericType" type="cct:NumericType"/>
    <xsd:simpleType name="NumericType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:CoreComponentTypeComponent>
                    <cdp:UID>CCT008</cdp:UID>
                    <cdp:CategoryCode>CCT</cdp:CategoryCode>
                    <cdp:DictionaryEntryName>Numeric. Type</cdp:DictionaryEntryName>
                    <cdp:Definition>Numeric information that is assigned or is determined by
calculation, counting, or sequencing. It does not require a unit of quantity or unit of
measure.</cdp:Definition>
                    <cdp:VersionData>
                        <cdp:VersionID>1.0</cdp:VersionID>
                        <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                    </cdp:VersionData>
                    <cdp:ObjectClass>Numeric</cdp:ObjectClass>
```

```
                    </cdp:CoreComponentTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:decimal"/>
        </xsd:simpleType>
        <!-- ===== CCT: QuantityType ===== -->
        <xsd:element name="Quantity" type="cct:QuantityType"/>
        <xsd:complexType name="QuantityType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:CoreComponentTypeComponent>
                        <cdp:UID>CCT009</cdp:UID>
                        <cdp:CategoryCode>CCT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Quantity. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A counted number of non-monetary units possibly including
fractions.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Quantity</cdp:ObjectClass>
                    </cdp:CoreComponentTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:decimal">
                    <xsd:attribute name="unitCode" type="xsd:token" use="optional"/>
                    <xsd:attribute name="unitCodeListID" type="xsd:token" use="optional"/>
                    <xsd:attribute name="unitCodeListAgencyID" type="xsd:token" use="optional"/>
                    <xsd:attribute name="unitCodeListAgencyName" type="xsd:token" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!-- ===== CCT: TextType ===== -->
        <xsd:element name="Text" type="cct:TextType"/>
        <xsd:complexType name="TextType">
            <xsd:annotation>
                <xsd:documentation>
                    <cdp:CoreComponentTypeComponent>
                        <cdp:UID>CCT007</cdp:UID>
                        <cdp:CategoryCode>CCT</cdp:CategoryCode>
                        <cdp:DictionaryEntryName>Text. Type</cdp:DictionaryEntryName>
                        <cdp:Definition>A character string (i.e. a finite set of characters) generally in the
form of words of a language.</cdp:Definition>
                        <cdp:VersionData>
                            <cdp:VersionID>1.0</cdp:VersionID>
                            <cdp:ChangeDescription>Initial</cdp:ChangeDescription>
                        </cdp:VersionData>
                        <cdp:ObjectClass>Text</cdp:ObjectClass>
                    </cdp:CoreComponentTypeComponent>
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="languageID" type="xsd:language" use="optional"/>
                    <xsd:attribute name="languageLocaleID" type="xsd:token" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
</xsd:schema>
```

1.1.9 DON-Enterprise-CodeList-TemperatureMeasureUnitCode-1.0.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This code list schema module was copied, altered and reused by the DON NDR
Team as part of the StrikePlan example. The final code list standard will be
different from what is done in this module. It is used to illustrate how Code List modules define
simpleTypes, complexTypes and global element reused in the DON BIE Schema module.
Documentation according to the DON Component Documentation Paramenters Schema has not
been added. -->
<xsd:schema
targetNamespace="urn:us:gov:dod:don:enterprise:code:temperatureMeasureUnitCode:1:0"
xmlns:uqt1-0="urn:us:gov:dod:don:enterprise:udt:1:0" xmlns:tmuc1-
0b="urn:us:gov:dod:don:enterprise:code:temperatureMeasureUnitCode:1:0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0.0">
    <xsd:import namespace="urn:us:gov:dod:don:enterprise:udt:1:0" schemaLocation="DON-
Enterprise-UDT-1.0.0.xsd"/>
    <xsd:element name="TemperatureMeasureUnitCountryCode" type="tmuc1-
0b:TemperatureMeasureUnitCodeType"/>
    <xsd:complexType name="TemperatureMeasureUnitCodeType">
        <xsd:annotation>
            <xsd:documentation>
                <cdp:Instance>
                    <!-- Data and values stored in this space are meant for instance-processing
purposes, and are non-normative. -->
                    <cdp:Prefix>cnt</cdp:Prefix>
                    <cdp:CodeListQualifier>ISO 3166-1</cdp:CodeListQualifier>
                    <cdp:CodeListAgency>6</cdp:CodeListAgency>
                    <cdp:CodeListVersion>0.3</cdp:CodeListVersion>
                </cdp:Instance>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction base="uqt1-0:CodeType">
                <xsd:enumeration value="DEGF"/>
                <xsd:enumeration value="DEGC"/>
                <xsd:enumeration value="DEGK"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name="TemperatureMeasureUnitCode">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="DEGC"/>
            <xsd:enumeration value="DECF"/>
            <xsd:enumeration value="DEGK"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

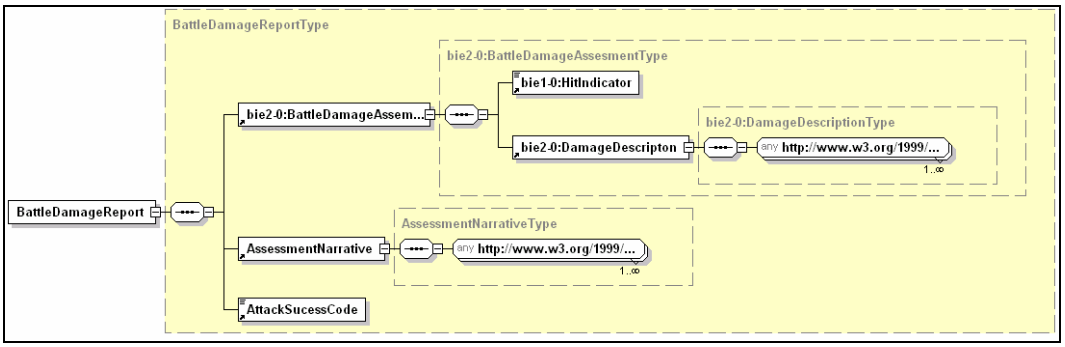
1.1.10 ISO3166.1-CodeList-CountryIdentificationCode-1.0.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:iso:codelist:3166-1:CountryIdentificationCode:1:0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:iso3166-1="urn:iso:codelist:3166-
1:CountryIdentificationCode:1:0" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1:0-beta">
    <xsd:element name="ISO31661CountryCode" type="iso3166-1:ISO31661CountryCodeType"/>
    <xsd:complexType name="ISO31661CountryCodeType">
        <xsd:annotation>
```

```
            <xsd:documentation>
                <ccts:Instance>
                    <!-- Data and values stored in this space are meant for instance-processing
purposes, and are non-normative. -->
                    <ccts:Prefix>cnt</ccts:Prefix>
                    <ccts:CodeListQualifier>ISO 3166-1</ccts:CodeListQualifier>
                    <ccts:CodeListAgency>6</ccts:CodeListAgency>
                    <ccts:CodeListVersion>0.3</ccts:CodeListVersion>
                </ccts:Instance>
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:normalizedString"/>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:schema>
```

1.2 Development Namespace Example: SPAWAR Battle Damage Report

1.2.1 Battle Damage Report Schema Model



1.2.2 BattleDamageReport.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<BattleDamageReport xmlns:bie1-0="urn:us:gov:dod:don:enterprise:bie:1:0" xmlns:bie2-
0="urn:us:gov:dod:don:enterprise:bie:2:0" xmlns:xhtm="http://www.w3.org/1999/xhtml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SPAWAR-BattleDamageReportRoot.xsd">
    <bie2-0:BattleDamageAssement>
        <bie1-0:HitIndicator>true</bie1-0:HitIndicator>
        <bie2-0:DamageDescripton>
            <xhtm:p>Commander's Assessement paragraph of content</xhtm:p>
        </bie2-0:DamageDescripton>
    </bie2-0:BattleDamageAssement>
    <AssessmentNarrative>
        <xhtm:p>Commander's Assessement paragraph of content</xhtm:p>
        <xhtm:ul>
            <xhtm:li>
                <xhtm:b>Scored a direct hit!</xhtm:b>
            </xhtm:li>
        </xhtm:ul>
    </AssessmentNarrative>
    <AttackSucessCode>DH</AttackSucessCode>
</BattleDamageReport>
```

1.2.3 SPAWAR-BattleDamageReportRoot.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is an example developmental Root-level Schema Template created for the DON NDRs. It
has all the necessary namespace declarations and imports based on the existence of namespace
versions per the example package provided with the NDRs. An Actual template will be provided in
conjunction with the NDR release that reference final component Schemas by resolvable URLs.
The StrikePlan example zip file must be unzipped in the exact directory structure provided inorder
for an XML Schema editor to be able to employ this schema template. -->
<!-- Rename this Schema using the following pattern: OrganizationName-Developmental-
ProcessOrDocumentName+"Root.xsd"-->
<!-- Note: Positions 6-9, and 10-12 of the targetNamespace URN are placeholders. Organizations
will need to substitute appropriate tokens as appropriate according to the DON NDR [NMS] rules.--
>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:bie2-
0="urn:us:gov:dod:don:enterprise:bie:2:0" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0-b">
```

```
    <!--  Note: The schema element has namespace declarations for all possible components. Instances not requiring components from a particular namespace may remove the xmlns attribute. Substitute and actual version number for "DevelopmentalSchemaVersion"-->
    <!-- Note: The xsd:import below makes available all global elements from the latest DON Enteprise BIE Reusable Schema module -->
    <xsd:import namespace="urn:us:gov:dod:don:enterprise:bie:2:0" schemaLocation="..\DON-Enterprise\DON-Enterprise-2.0\DON-Enterprise-BIE-Reusable-2.0.1.xsd"/>
    <!-- Note: The xsd:include statment below makes available global elements defined in the no-namespace DevlopmentalBIEReusable.xsd module. Substitue an abbreviation for  "OrgX" representing our organization's name. -->
    <!-- Note: Be sure to change the schemaLocation attribute value to the actual name chosen for the Developmental BIE Schema module. -->
    <xsd:include schemaLocation="SPAWAR-Developmental-BIE.xsd"/>
    <xsd:element name="BattleDamageReport" type="BattleDamageReportType">
        <xsd:annotation>
            <xsd:documentation>This element MUST be conveyed as the root element in any instance document based on this Schema expression.</xsd:documentation>
        </xsd:annotation>
        <!-- Root-level global element declaration. For Schemas use for content publication, there can be only one Root-level element [RED1]. For Schema describing business process transactions, there may be multiple Root-level elements, one for each step in a process [RED2]. -->
    </xsd:element>
    <xsd:complexType name="BattleDamageReportType">
        <xsd:sequence>
            <xsd:element ref="bie2-0:BattleDamageAssement"/>
            <xsd:element ref="AssessmentNarrative"/>
            <xsd:element ref="AttackSucessCode"/>
            <!-- Insert references to global elements defined in DON-Enterprise-BIE-Reusable.xsd, or no-namespace DevelopmentalReusable.xsd file or nested xsd:sequence/xsd:choice elements.-->
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

1.2.4 SPAWAR-DevelopmentalBIE.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is an example BIE Reusables Schema module that contains development CCTS BIE XML Schema complexTypes and Global Elements. It has no targetNamespace, and is therefore included in Developmental Root-level Schema modules. Harmonized types and global elements created in this module will be moved from this module to a module in the enterprise namespace.-->
<xsd:schema xmlns:qdt1-0="urn:us:gov:dod:don:enterprise:qdt:1:0" xmlns:dqdt="urn:us:gov:dod:don:spawar:qdt" xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
    <xsd:import namespace="urn:us:gov:dod:don:enterprise:bie:2:0" schemaLocation="..\DON-Enterprise\DON-Enterprise-2.0\DON-Enterprise-BIE-Reusable-2.0.1.xsd"/>
    <!-- Note: The following xsd:import makes available developmental Qualified Data Types created in the Developmental QDT XSD. xsd:complexTypes representing CCTS Qualified Data Types are developed in a namespace assigned schema module because their names are frequently the same as the names of xsd:complexTypes in no-namespace Developmental BIE XSDs.-->
    <!-- Warning: Be sure to change the value of the schemaLocation attribute to match the actual name chosen for the Developmental QDT Schema module.-->
    <xsd:import namespace="urn:us:gov:dod:don:spawar:qdt" schemaLocation="SPAWAR-Developmental-QDT.xsd"/>
    <!--==== Basic BIE Property ComplexTypes and Global Elements ===-->
    <xsd:element name="AttackSucessCode" type="AttackSuccessCodeType"/>
    <xsd:complexType name="AttackSuccessCodeType">
        <xsd:simpleContent>
            <xsd:extension base="dqdt:AttackSuccessCodeType"/>
```

```
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:element name="AssessmentNarrative" type="AssessmentNarrativeType"/>
    <xsd:complexType name="AssessmentNarrativeType">
        <xsd:complexContent>
            <xsd:extension base="qdt1-0:XHTMLContentType"/>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

1.2.5 SPAWAR-DevelopmentalQDT.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is an example BIE Reusables Schema module that contains development CCTS BIE XML Schema complexTypes and Global Elements. It has no targetNamespace, and is therefore included in Developmental Root-level Schema modules. Harmonized types and global elements created in this module will be moved from this module to a module in the enterprise namespace.-->
<xsd:schema targetNamespace="urn:us:gov:dod:don:spawar:qdt" xmlns:udt1-0="urn:us:gov:dod:don:enterprise:udt:1:0" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:us:gov:dod:don:spawar:qdt" elementFormDefault="qualified" attributeFormDefault="unqualified" version="SchemaVersion">
    <!--  Note: The following import is by convention the highest Schema version number in the highest DON Enterprise BIE namespace. The Schema modularity and versioning strategy is specifically designed to ensure that by importing this namespace and Schema, your developmental components have access to and can be based on the latest versions of all enteprise BIE, Qualified DataType and Unqualified DataType XLM Schema compoents. -->
    <xsd:import namespace="urn:us:gov:dod:don:enterprise:qdt:1:1" schemaLocation="..\DON-Enterprise\DON-Enteprise-1.1\DON-Enterprise-QDT-1.1.1.xsd"/>
    <!--========== Qualified Data Type Complex Types ===============-->
    <!--*** Data Type Attack Sucess_ Code. Type-->
    <xsd:complexType name="AttackSuccessCodeType">
        <xsd:simpleContent>
            <xsd:restriction base="udt1-0:CodeType">
                <xsd:pattern value="DH"/>
                <xsd:pattern value="IH"/>
                <xsd:pattern value="MS"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <!--*** Strike_ Identifier. Type ***-->
    <xsd:complexType name="StrikeIDType">
        <xsd:simpleContent>
            <xsd:restriction base="udt1-0:IdentifierType">
                <xsd:pattern value="\d{8}-[A-Z]-\d{2}"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:schema>
```

Appendix E
Approved XML Specifications

Frameworks

RosettaNet

RNIF Specification V02.00.01

The RNIF V02.00.01 (.zip) contains the RNIF Specification, three Headers (Preamble, Delivery & Service Header) and two Business Signals (Receipt Acknowledgement and Exception).

ISO

ISO 11179–Information Technology—Metadata Registries

ISO/IEC 11179-1:1999 Information technology—Specification and standardization of data elements—Part 1: Framework for the specification and standardization of data elements (available in English only)

ISO/IEC 11179-2:2000 Information technology—Specification and standardization of data elements—**Part 2:** Classification for data elements (available in English only)

ISO/IEC 11179-3:2003 Information technology—Metadata registries (MDR)—Part 3: Registry metamodel and basic attributes (available in English only)

ISO/IEC 11179-4:2004 Information technology—Metadata registries (MDR)—Part 4: Formulation of data definitions (available in English only)

- **ISO/IEC 11179-5:1995** Information technology—Specification and standardization of data elements—Part 5: Naming and identification principles for data elements (available in English only)

ISO/IEC 11179-6:1997 Information technology—Specification and standardization of data elements—Part 6: Registration of data elements (available in English only)

- ISO/TS 15000-1:2004 Electronic business eXtensible Markup Language (ebXML)—**Part 1:** Collaboration-protocol profile and agreement specification (ebCPP) (available in English only)

- ISO/TS 15000-2:2004 Electronic business eXtensible Markup Language (ebXML)—**Part 2:** Message service specification (ebMS) (available in English only)

- ISO/TS 15000-3:2004 Electronic business eXtensible Markup Language (ebXML)—**Part 3:** Registry information model specification (ebRIM) (available in English only)

- ISO/TS 15000-4:2004 Electronic business eXtensible Markup Language (ebXML)—**Part 4:** Registry services specification (ebRS) (available in English only)

ISO 15000-5 ebXML Core Components Technical Specification (Candidate - Out for vote)

OASIS

DocBook v4.1 January 2001

The DocBook V4.1 OASIS Standard is available as a zip archive.

Directory Services Markup Language (DSML) v2.0 January 2002

The DSML v2 OASIS Standard

ebXML Collaborative Partner Profile Agreement (CPPA) v2 June 2002

The OASIS ebXML CPPA v2.0 OASIS Standard (Also published as ISO 15000-1)

ebXML Message Service Specification v2.0 April 2004

The OASIS ebXML Message Service Specification v2.0 OASIS Standard (Also published as ISO 15000-2)

ebXML Registry Services v2.5 February 2002

The OASIS ebXML Registry Services v2.0 OASIS Standard

ebXML Registry Information Model (RIM) v2.0 March 2002

The OASIS ebXML Registry Information Model v2.0 OASIS Standard

Security Assertion Markup Language (SAML) v1.1 August 2003

The complete SAML V1.1 OASIS Standard set (PDF format) and schema files are available in this zip file.

UDDI v2.0 February 2003

- UDDI Version 2 API Specification. UDDI Version 2.04 API, Published Specification, Dated 19 July 2002: HTML/PDF (492 KB)
- UDDI Version 2 Data Structure. UDDI Version 2.03, Data Structure Reference, Published Specification, Dated 19 July 2002: HTML/PDF (349 KB)
- UDDI Version 2 XML Schema. Version 2.0 UDDI XML Schema 2001: uddi_v2.xsd
- UDDI Version 2 Replication UDDI Version 2.03, Replication Specification, Published Specification, Dated 19 July 2002: HTML/PDF (251 KB)
- UDDI Version 2 XML Replication Schema. Version 2.03 Replication XML Schema 2001: uddi_v2replication.xsd
- UDDI Version 2 XML Custody Schema. UDDI XML Custody Schema: uddi_v2custody.xsd
- UDDI Version 2 Operator's Specification. UDDI Version 2.01, Operator's Specification, Published Specification, Dated 19 July 2002: HTML/PDF (205 KB)
- UDDI Version 2 WSDL Service Interface Descriptions. UDDI Inquire API: inquire_v2.wsdl and UDDI Publish API: publish_v2.wsdl
- UDDI Version 2 tModels:
  UDDI Registry tModels
  UDDI Other Core tModel
  Replication tModels
  Taxonomy tModels

Universal Business Language (1p0)

UBL 1.0 Committee Draft is located at http://docs.oasis-open.org/ubl/cd-UBL-1.0/

The compressed archive containing the entire release is available for local installation fromhttp://docs.oasis-open.org/ubl/cd-UBL-1.0.zip

Web Services for Remote Portlets (WSRP) v1.0 April 2003

The WSRP v1.0 OASIS Standard

UN/CEFACT

UN/CEFACT Modeling Methodology

UN/CEFACT–UN/CEFACT, Core Components Technical Specification: Part 8 of the ebXML Framework, 15 November 2003, Version 2.01.

W3C

Extensible Markup Language (XML) 1.0 (Third Edition)
First published 10 February 1998, revised 4 February 2004, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau - (Errata)

Namespaces in XML
14 January 1999, Tim Bray, Dave Hollander, Andrew Layman - (Errata)

XML Schema Part 0: Primer
2 May 2001, David C. Fallside - (Errata)

XML Schema Part 1: Structures
2 May 2001, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn - (Errata)

XML Schema Part 2: Datatypes
2 May 2001, Paul V. Biron, Ashok Malhotra - (Errata)

XML Encryption Syntax and Processing
10 December 2002, Donald Eastlake, Joseph Reagle - (Errata)

Decryption Transform for XML Signature
10 December 2002, Merlin Hughes, Takeshi Imamura, Hiroshi Maruyama - (Errata)

XML-Signature Syntax and Processing
12 February 2002, Donald Eastlake, Joseph Reagle, David Solo - (Errata)

XML-Signature XPath Filter 2.0
8 November 2002, John Boyer, Merlin Hughes, Joseph Reagle - (Errata)

XML Path Language (XPath) Version 1.0
16 November 1999, James Clark, Steven DeRose - (Errata)

Extensible Stylesheet Language (XSL) Version 1.0
15 October 2001, Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham, Paul Grosso, Eduardo Gutentag, R. Alexander Milowski, Scott Parnell, Jeremy Richman, Steve Zilles - (Errata)

XSL Transformations (XSLT) Version 1.0
16 November 1999, James Clark - (Errata)

XML Linking Language (XLink) Version 1.0
27 June 2001, Steven DeRose, Eve Maler, David Orchard - (Errata)

XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)
First published 26 January 2000, revised 1 August 2002, Steven Pemberton - (Errata)

Cascading Style Sheets, level 2 (CSS2) Specification
12 May 1998, Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs - (Errata)

SOAP Version 1.2 Part 0: Primer
24 June 2003, Nilo Mitra - (Errata)

SOAP Version 1.2 Part 1: Messaging Framework
24 June 2003, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Martin Gudgin - (Errata)

SOAP Version 1.2 Part 2: Adjuncts
24 June 2003, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Martin Gudgin, Marc Hadley, Noah Mendelsohn - (Errata)

SOAP Version 1.2 Specification Assertions and Test Collection
24 June 2003, Oisin Hurley, Anish Karmarkar, Jeff Mischkinsky, Mark Jones, Lynne Thompson, Richard Martin, Hugo Haas - (Errata)

Scalable Vector Graphics (SVG) 1.1 Specification
First published 4 September 2001, revised 14 January 2003, Jon Ferraiolo, 藤沢 淳, Dean Jackson - (Errata)

Document Object Model (DOM) Level 3 Core Specification
7 April 2004, Jonathan Robie, Mike Champion, Steve Byrne, Arnaud Le Hors, Philippe Le Hégaret, Lauren Wood, Gavin Nicol - (Errata)

Document Object Model (DOM) Level 3 Load and Save Specification
7 April 2004, Johnny Stenback, Andy Heninger - (Errata)

Resource Description Framework (RDF): Concepts and Abstract Syntax
10 February 2004, Graham Klyne, Jeremy J. Carroll - (Errata)

RDF Semantics
10 February 2004, Patrick Hayes - (Errata)

RDF Primer
10 February 2004, Frank Manola, Eric Miller - (Errata)

This specification supersedes Resource Description Framework (RDF) Model and Syntax Specification published on 22 February 1999

RDF Vocabulary Description Language 1.0: RDF Schema
10 February 2004, Dan Brickley, Ramanathan V. Guha - (Errata)

RDF/XML Syntax Specification (Revised)
10 February 2004, Dave Beckett - (Errata)

RDF Test Cases
10 February 2004, Jan Grant, Dave Beckett - (Errata)

OWL Web Ontology Language Use Cases and Requirements
10 February 2004, Jeff Heflin - (Errata)

OWL Web Ontology Language Overview
10 February 2004, Deborah L. McGuinness, Frank van Harmelen - (Errata)

OWL Web Ontology Language Guide
10 February 2004, Michael K. Smith, Chris Welty, Deborah L. McGuinness - (Errata)

OWL Web Ontology Language Reference
10 February 2004, Guus Schreiber, Mike Dean - (Errata)

OWL Web Ontology Language Semantics and Abstract Syntax
10 February 2004, Peter F. Patel-Schneider, Patrick Hayes, Ian Horrocks - (Errata)

OWL Web Ontology Language Test Cases
10 February 2004, Jeremy J. Carroll, Jos De Roo - (Errata)

XForms 1.0
14 October 2003, Micah Dubinko, Leigh L. Klotz, Roland Merrick, T. V. Raman - (Errata)

Other

Simple API for XML (SAX) v1.0 & v2.0 May 11, 1998


XML
DEPARTMENT OF THE NAVY