FEDERAL XML NAMING AND DESIGN RULES

TABLE OF CONTENTS

1 Introduction

1.1 AUDIENCES

The document is primarily intended for developers already familiar with XML. This document focuses on XML Schemas as a tool for interoperability. To realize the maximum benefit, we suggest that you take the time to become familiar with the XML Schema language.

We encourage developers to try the techniques recommended here and provide feedback via the Federal CIO Council XML Working Group listserv[1]. We will collect lessons learned and best practices, updating and expanding this document periodically.

1.2 SCOPE

This Federal XML Naming and Design Rules document is applicable to all Executive Branch Departments and Agencies (hereinafter referred to as Agency) that use XML, including all commercial and government off-the-shelf XML related product implementations. It is applicable to all contractors and vendors doing XML development work on behalf of Departments and Agencies.

1.3 TERMINOLOGY AND NOTATION

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

[Definition] – A formal definition of a term. Definitions are normative.

[Example] – A representation of a definition or a rule. Examples are informative.

[Note] – Explanatory information. Notes are informative.

[1] You can subscribe to the listserv from the following site: http://www.xml.gov/working_group.htm.

[RRR*n*] - Identification of a rule that requires conformance to ensure that an XML Schema is conformant. The value RRR is a prefix to categorize the type of rule where the value of RRR is as defined in Table 1 and n (1..n) indicates the sequential number of the rule within its category. In order to ensure continuity across versions of the specification, rule numbers that are deleted in future versions will not be re-issued, 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.

Figure 1 - Rule Prefix Token Value

Rule Prefix Token	Value
ATD	Attribute Declaration
ATN	Attribute Naming
CDL	Code List
CTD	ComplexType Definition
DEN	Data Element Dictionary Entry Names and Definitions
DOC	Documentation
ELD	Element Declaration
ELN	Element Naming
GNR	General Naming
GTD	General Type Definition
GXS	General XML Schema
IND	Instance Document
MDC	Modeling Constraints

NMC	Naming Constraints
NMS	Namespace
RED	Root Element Declaration
SSM	Schema Structure Modularity
STD	SimpleType Definition
STR	Standards Requirements
VER	Versioning

Bold - The bolding of words is used to represent example names or parts of names taken from the library.

`Courier` – All words appearing in **`courier font`** are values, objects, and keywords.

Italics – All words appearing in italics, when not titles or used for emphasis, are special terms defined in Appendix A.

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 World Wide Web Consortium (W3C) *XML Schema Definition Language* (XSD) Recommendations. See Appendix A for additional term definitions.

1.4 CONFORMANCE TO PL 104-113 AND OMB A119

Public Law 104-113 requires the use of voluntary consensus standards where appropriate rather than development of proprietary standards. Office of Management and Budget (OMB) Circular A-119 Authorizes Agencies to expend funds in support of participating in the development of standards.

STR1	To ensure conformance with both statutory and policy requirements contained in Public Law 104-113 and Office of Management and Budget Circular A-119, all Federal XML implementations must adhere to the following hierarchy of standards in creating and using XML

* De jure Voluntary Consensus Standards * Cross-sector Voluntary Consensus Standards * Sector specific Voluntary Consensus Standards * Federal Enterprise Wide Standards * Agency specific standards

Agencies are encouraged to address XML at an enterprise level.

SR2	Agencies SHOULD create Agency level policy, procedures and guidance to ensure XML is developed and governed at an enterprise level

As Agencies move forward with enterprise level approaches to XML, Agency level XML components – elements, complex types, and schema will begin to emerge. These components provide significant wealth to the federal government as a whole and should be made available for reuse. The best way to make this happen is for Agencies to promote their XML components as candidate Federal components, and to appropriate Voluntary Consensus Standards bodies.

STR3	Agencies SHOULD promote Agency level XML components to candidate federal level components and candidate Voluntary Consensus Standards Bodies

1.4.1 W3C Recommendations

In keeping with the basic tenants of PL 104-113 and OMB A-119, Federal XML will be based on recognized standards rather than developing proprietary standards. Standards promulgated by nationally or internationally accredited standards bodies (such as ISO, IEEE, ANSI, UN/CEFACT, etc.)should always be adhered to when developing applications within the domain that the standard addresses. The only exception to this is when a standard produced by one of these bodies competes with a similar product of the W3C. In this case, only, the W3C should have precedence. Although competing standards exist in the XML space, the World Wide Web Consortium is the only International Standards organization that is committed to ensuring end-to-end interoperability of its suite of XML standards. The W3C standards are recognized throughout the world as the authoritative set of XML standards, and the vast majority of developers use W3C standards rather than those of other competing bodies. Accordingly, Federal XML will be primarily based on the W3C suite of XML standards.

[STA1]	All schema design rules MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

In general, production applications should only use software that implements W3C Final Recommendations and final specifications of the accredited standards bodies referenced in the above paragraph.

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

Applications using software that implement W3C technical reports at other stages of the development process, or other draft products of Voluntary Consensus Standards bodies, should only do so with the following restrictions:

- Production Applications:
  - Prior to creating, incorporating or using software that implements non-W3C specifications, activities MUST:
    - Ensure that no competing W3C endorsed recommendation exists or is being developed.
    - Ensure that the specification is a product of an accredited standards body (ISO, IEEE, ANSI, UN/CEFACT) or a credible Voluntary Consensus Standards body such as the Organization for the Advancement of Structured Information Standards (OASIS), the OMG, OAG, UDDI, RosettaNet, or BizTalk.
  - Activities MAY choose to implement W3C technical reports with a Proposed Recommendation status, provided they are committed to immediately updating software should any changes be made when the report reaches final status
- Pilot Applications: Activities developing pilot applications (as a precursor to production) MAY also implement software that conforms to W3C technical reports with a Candidate Recommendation status.
- (Advanced Concept) Demonstrations: Activities developing demonstration applications (as a proof of concept) MAY also implement software that conforms to W3C technical reports with a Working Draft status or another accredited standards body or Voluntary Consensus Standards body's draft specifications.
  - Exception:
  - Activities MAY implement the SAX 1.0 and 2.0.

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

Proprietary extensions to W3C Technical Specifications or other specifications by accredited standards bodies should not be employed in any software or XML document (instance, schema, stylesheet) that will be shared publicly with activities outside a particular development environment.

[STA3]	Proprietary extensions to the W3C specifications MUST never be used.

1.4.2 OASIS Standards

1.5 GUIDING PRINCIPLES

Federal XML guiding principles encompass three areas:

- General guiding principles
- Extensibility
- Code generation

1.5.1 General Guiding Principles

- Internet Use – Federal XML implementations shall be straightforwardly usable over the Internet.
- Tool Use and Support – The design of Federal XML will not make any assumptions about sophisticated tools for creation, management, storage, or presentation being available, nor levy any government unique requirements that would necessitate tool vendors developing proprietary extensions to support Federal requirements.
- Legibility – Federal XML documents should be human-readable and reasonably clear.

- Simplicity – Schema design must be as simple as possible (but no simpler).

- Component Reuse – Federal schema should contain as many common features as possible.

- Standardization – The number of ways to express the same information in a Federal XML instance is to be kept as close to one as possible.

- Customization and Maintenance – Federal schema must facilitate customization and maintenance.

- Prescriptiveness – Federal schema design will balance prescriptiveness in any single usage scenario with prescriptiveness across the breadth of usage scenarios supported. Having precise, tight content models and Datatypes is a good thing.

- XML Technology – Federal schema design will avail itself of standard XML processing technology wherever possible (XML itself, XML Schema, XSLT, XPath, and so on). Federal XML implementations will be cautious about basing decisions on "standards" (foundational or vocabulary) that are works in progress.

- Legacy formats – Federal schema design is not responsible for catering to legacy formats or syntaxes.

1.5.2 Design For Extensibility

Many e-commerce document types are, broadly speaking, useful but require minor structural modifications for specific tasks or markets. When a truly common XML structure is to be established for e-commerce, it needs to be easy and inexpensive to modify.

Many data structures used in e-commerce are very similar to "standard" data structures, but have some significant semantic difference native to a particular industry or process. In traditional Electronic Data Interchange (EDI), there has been a gradual increase in the number of published components to accommodate market-specific variations. Handling these variations are a requirement, and one that is not easy to meet. A related EDI phenomenon is the overloading of the meaning and use of existing elements, which greatly complicates interoperation.

To avoid the high degree of cross-application coordination required to handle structural variations common to EDI and XML Document Type Definition (DTD)

based systems - it is necessary to accommodate the required variations in basic data structures without either overloading the meaning and use of existing data elements, or requiring wholesale addition of new data elements. This can be accomplished by allowing implementers to specify new element types that inherit the properties of existing elements, and to also specify exactly the structural and data content of the modifications.

This can be expressed by saying that extensions of core elements are driven by context.[3] Context driven extensions should be renamed to distinguish them from their parents, and designed so that only the new elements require new processing.

Similarly, data structures should be designed so that processes can be easily engineered to ignore additions that are not needed.

1.5.3 Data versus Document Centric XML

XML as an web-enabled subset of SGML, was initially envisioned as strictly for document centric (publishing) application. However, the value of XML in the data centric world quickly became apparent. The advent of the XML Schema Definition Language (XSD) and it's strong datatyping capabilities have enabled robust data-centric applications that now outnumber document centric approaches. XSD is useable for both data centric and document centric solutions. It is essential that Federal XML be as uniform as possible for both pieces of the XML solution set so that XML Schema – regardless of which focus they have – are as consistent as possible. Federal XML NDRs will be selected with this objective in mind.

1.5.4 Code Generation

This NDR document makes no assumptions on the availability or capabilities of tools to generate Federal conformant XSD Schemas. The Federal NDR design process has scrupulously avoided establishing any naming or design rules that sub-optimizes the XSD in favor of tool generation. Conformance to W3C technical specifications holding recommended status ensures that no favoritism is being given to particular implementation approaches or tools in generating XML.

---

[3] ebXML, Core Components Technical Specification – Part 8 of the ebXML Technical Framework, V2.0, 11 August 2003

1.6 DOCUMENT ORGANIZATION

This Federal Naming and Design Rules document is organized as follows:

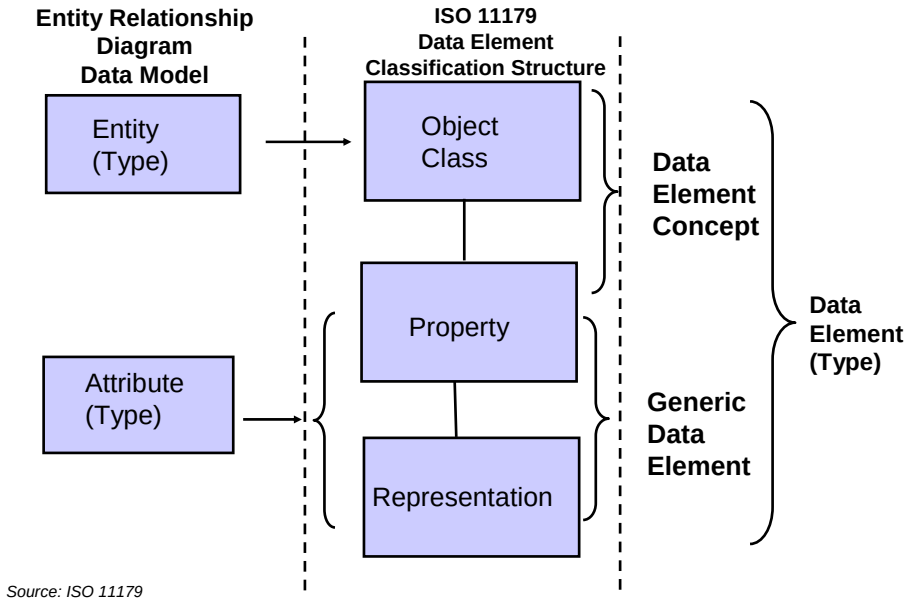
2 Information Analysis

Sound schema design, especially for schema supporting data centric XML exchanges, must have a solid foundation in sound data models. Commonality in data modeling, and by extension process modeling, enables the expression of schema naming and design rules in a consistent fashion.

2.1 DEFINING DATA

Data Models are typically defined in terms of Entity Relationship Diagrams or Data Classification Structures (Class Diagrams). As shown in Figure 2-1, these two models share common concepts. Both entities and objects represent discrete things. Both entities and objects have data element concepts that identify that portion of a data element that is independent of any particular data typing or representation. Both entities and objects have simple data elements, that part of a data element that describes its characteristic. The fundamental difference is that entities only define the data, whereas class diagrams define both the data and its behavior.

Figure 2-1. Data Structures



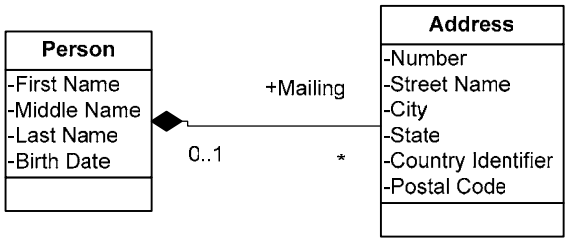
In an entity relationship data model, each entity may have multiple attributes. In a class based data model, each Class may have multiple properties. The element that represents an Entity with all of its attributes, or a class with all of its properties is an aggregation (complex) data element. Complex data elements can have varying contents depending on the way in which it is being used. Data elements that represent an attribute of an entity, or a property plus representation of a type, represent simple (generic) data elements. Simple data elements are reusable across multiple entities/classes.

2.2 RELATIONSHIPS BETWEEN DATA ELEMENTS

Complex data elements may use other complex data elements as one of their properties. For example, a *Person* complex data element may use a *Address* complex data element. In this relationship, *Person* has an association with *Address*. *Person* is said to inherit the properties of *Address*.

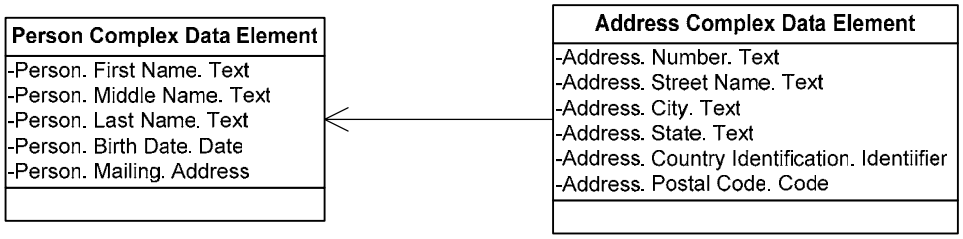
In a UML diagram, such as figure 2-2, a physical connection is made between the classes and the nature of that relationship is expressed as a word or words that reflect the role of the associated class as the property of the associating class. Figure 2-3 indicates that the association between Person and Address is Mailing Address.

Figure 2-2 Class Association in UML



In a semantic diagram, such as figure 2-4, the properties of the associated class represented by the complex data element *Mailing* are represented as a single property in the associating class *Person as Person. Mailing. Addess.*.

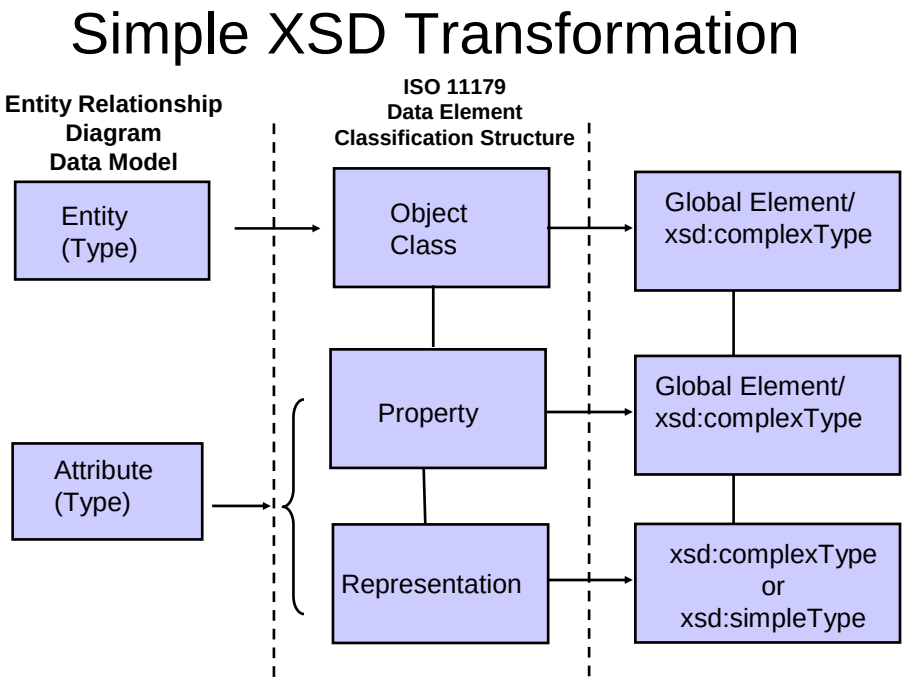
Figure 2-3 Class Association in Semantic Diagram



2.3 TRANSFORMING DATA INTO XML

Entity relationship diagrams, object class models, and semantic diagrams express conceptual data models that enable transforming their data constructs into consistent, sound XSD artifacts. As shown in Figure 2-4, the complex data elements that represent entities or object classes can be expressed as global elements and named complex types. Attributes and properties that represent simple data element properties can also be expressed as global elements and complex types. Representation terms as the expression of the data type, can be expressed as either complex or simple types depending on their usage requirements.

Figure 2-4. Transforming Data constructs into XML Artifacts



Beyond the expression of simple and complex data elements, we must also accommodate the relationships (associations) between classes. This is accomplished by declaring global elements for the association, which then, just as with the semantic model shown in Figure 2-4, become part of the content model of the associating (inheriting) xsd:complexType.

3 General XML Constructs

3.1 Overall Schema Structure

Consistency in development of schema includes consistency in the appearance of schema. By defining a canonical form for Federal schema, Agency Schema will all have the same look and feel, which will result in reduced developer's costs.

[GXS1] Data-centric Schema MUST conform to the following physical layout as applicable:

XML Declaration

<!-- ===== Copyright Notice ===== -->

Any applicable copyright notice

<!-- ===== xsd:schema Element With Namespaces Declarations ===== -->

xsd:schema element to include version attribute and namespace declarations in the following order:

xmlns:xsd

Target namespace

Default namespace

CommonComplexElements

CommonSimpleElements

Datatypes

Identifier Schemes

Code Lists

Attribute Declarations – elementFormDefault="qualified" attributeFormDefault="unqualified"

<!-- ===== Imports ===== -->

CommonComplexElements schema module(s)

CommonSimpleElements schema module(s)

Unqualified Datatypes schema module(s)

Qualified Datatypes schema module(s)

Code and Identifier List schema module(s)

<!-- ===== Global Attributes ===== -->

Global Attributes and Attribute Groups

<!-- ===== Root Element ===== -->

Root Element Declaration

Root Element Type Definition

<!-- ===== Element Declarations ===== -->

alphabetized order

<!-- ===== Type Definitions ===== -->

All type definitions segregated by simple and complex as follows

<!-- ===== Complex Data Element Type Definitions ===== -->

alphabetized order of Complex Data Element xsd:complexType definitions

<!-- =====Simple Data Element Type Definitions ===== -->

alphabetized order of simple data element xsd:complexType definitions

<!-- ===== Copyright Notice ===== -->

Required copyright notice.

3.1.1 Root Element

XML 1.0 describes the concept of a root element that serves as the anchor for an XML instance. Specifically, XML 1.0 states "There is exactly one element, called the **root**, or document element, no part of which appears in the content of any other element." The root element has many uses. The root element may or may not identify proper application space handling through an attribute. Specifically, XML 1.0 states "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."

Any global element can function as the instance root element. Since the Federal schema will have all elements declared globally, this has the potential to cause confusion amongst different developers and users. Accordingly, Federal schema will identify one global element as the root element for a particular instance. This will be accomplished through an `xsd:annotation` child element for that element in accordance with the following rule:

[ELD1]	Each `Schema` MUST identify one and only one global element declaration that defines the document level container being conveyed 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 "*`This element MUST be conveyed as the root element in any instance document based on this Schema expression.`*"

3.2 CONSTRAINTS

3.2.1 Naming Constraints

[NMC1]	Each dictionary entry name MUST define one and only one fully qualified path (FQP) for an element or attribute.

3.2.2 Modeling Constraints

[MDC1]	Libraries and Schemas MUST only use approved datatypes.

[MDC2]	Mixed content MUST NOT be used in data centric schema except where contained in an `xsd:documentation` element.

3.3 REUSABILITY SCHEME

[ELD2]	All data-centric element declarations MUST be global. All document-centric element declarations SHOULD be global.

3.4 NAMESPACE SCHEME

3.4.1 Declaring Namespaces

[NMS1]	Every schema module, except internal schema modules, MUST have a namespace declared using the `xsd:targetNamespace` attribute.

[NMS2]	Every defined or used schema set version MUST have its own unique namespace

[NMS3]	Federal Namespaces MUST only contain federally developed schema modules

[NMS7]	Published namespaces MUST never be changed.

3.4.2 Namespace Uniform Resource Indicators

I rcommend URNs due to persistence which is a higher overriding concern than resolvability (which is only important for schema location). I recommend we look at the Recommendations for Federal Namespaces paper and use the recommendations contained therein as modified to include more than just schema per a merger of the approach in that paper as well as the expanded UN/CEFACT URN scheme as shown in the following:

- 1st Level Domain – NID of US
- Second Level Domain – Organization Hierarchy. In this case GOV
- Third Level Domain – Specific Government Hierarchy – EPA, OMB, DoD, Treasury
- Fourth Level Domain – Agency Level Hierarchy – USN, USAF, IRS, FMS
- Fifth Level Domain – Resource Type – Schema or Other as Identified
- Sixth Level Domain – Resource Status
- Seventh Level Domain – Resource Name
- Eighth Level Domain - Versioning

[NMS4]	Agency Namespaces MUST only contain agency developed schema modules

[NMS5]	The namespace names for Schemas holding draft status MUST be of the form: `urn:us:`

[NMS6]	The namespace names for Schemas holding Approved status MUST be of the form: `urn:us:`

3.4.3 Persistence

[NMS7]	Published namespaces MUST never be changed.

3.5 VERSIONING SCHEME

3.5.1 Draft Schema

[VER1]	Every federal and Agency Schema and schema module major version committee draft MUST have a document-id of the form `<name>-<major>.0[.<revision>]`

[VER3]	Every minor version schema or schema module draft MUST have a document-id of the form `<name>-<major >.<non-zero>[.<revision>]`

3.5.2 Standard Schema

[VER2]	Every federal and Agency Schema and schema module major version Standard MUST have a document-id of the form `<name>-<major>.0`

[VER4]	Every minor version schema or schema module Standard MUST have an document-id of the form `<name>-<major >.<non-zero>`

3.5.3 Minor Version Changes

[VER5]	For minor version changes, the name of the version construct MUST NOT change.

[VER9]	Schema and schema module minor version changes MUST be limited to the use of xsd:extension or xsd:restriction to optionally alter existing types or add new constructs.

[VER10]	Schema and schema module minor version changes MUST not break semantic compatibility with prior versions.

3.5.4 Versioning Numbering Scheme

[VER6]	Every schema and schema module major version number MUST be a sequentially assigned, incremental number greater than zero.

[VER7]	Every schema and schema module minor version number MUST be a sequentially assigned, incremental non-negative integer.

3.5.5 Versioning Import Requirements

[VER8]	A minor version document schema MUST import its immediately preceding version document schema.

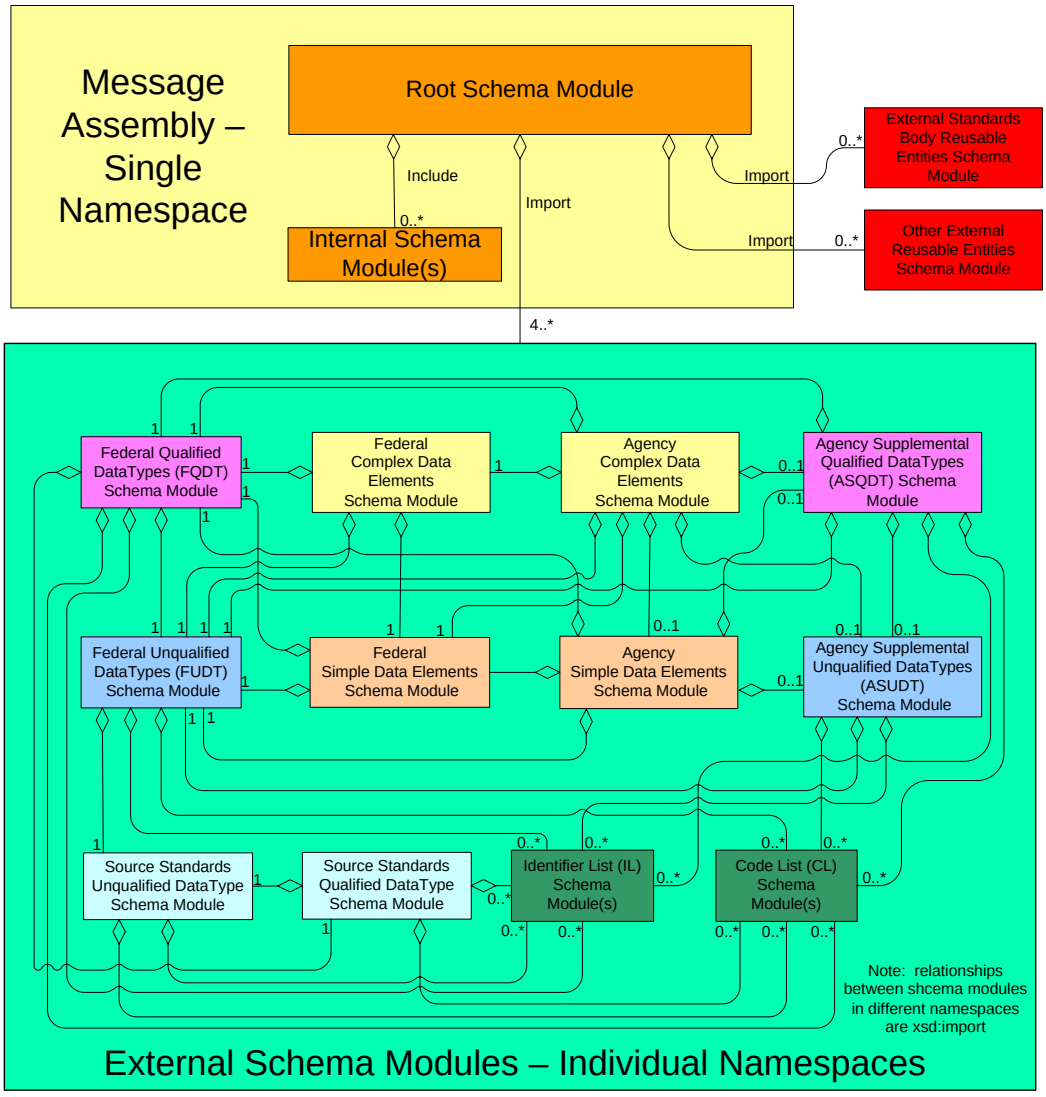
3.6 MODULARITY

A robust Federal modularity model must support:

- Common and Agency Unique Reusable components at the Data and Aggregation Levels
- Common and Agency Unique DataTypes
- Reusable Code and Identifier Lists
- Reusable State and Local modules
- Reusable External Standards Body modules

Figure 3-1 reflects such a model. The federal modularity model has a highly modularized structure that actively supports reuse and standardization. The federal modularity model provides for both federal and Agency level schema modules for complex data elements, simple data elements and datatypes; and also supports both internal and external schema modules for code lists and identifier lists.

Figure 3-1 Federal Schema Modularity Model



3.6.1 Leveraging VCS Datatypes

At the heart of the Federal schema modularity model are Voluntary Consensus Standard datatype schema that provide for consistent expression of the most common core datatypes typically used in information sharing. These VCS datatype schema serve as the basis of – are imported into – federal unqualified datatype schema. These federal unqualified datatypes have no restrictions placed on them. Additionally, as Agencies promote their own restricted datatypes to federal enterprise level standards, an additional federal qualified datatype schema will emerge. Both the federal unqualified and qualified datatype schema modules will serve as the basis of – are imported into – Agency unique unqualified and qualified schema modules.

3.6.2 Schema Modules

[SSM1]	Root Schema expressions MAY be split into multiple schema modules.

[SSM2]	A root schema in one namespace that is dependent upon type definitions or element declarations defined in another namespace MUST only import the root schema from that namespace.

[SSM3]	A root schema in one namespace that is dependant upon type definitions or element declarations defined in another namespace MUST NOT import internal schema modules from that namespace.

[SSM4]	All imported schema modules MUST be fully conformant with the Federal XML naming and design rules.

[SSM5]	Schema modules MUST either be treated as external schema modules or as internal schema modules of the root schema.

3.6.2.1 INTERNAL SCHEMA MODULES

[SSM6]	All internal schema modules MUST be in the same namespace as their corresponding root schema.

[SSM7]	Each internal schema module MUST be named `{ParentSchemaModuleName}{InternalSchemaModuleFunction}{schema module}`

3.6.3 Federal External Schema Modules

3.6.3.1 Complex Data Element Schema Modules

[SSM8]	A schema module defining all Federal `Common Complex Data Elements` MUST be created.

[SSM9]	The Federal Common Complex Data Elements schema module MUST be named "`fed:Common Complex Data Elements Schema Module`"

3.6.3.2 Common Simple Data Elements

[SSM10]	A schema module defining all Federal `Common Simple Data Elements` MUST be created.

[SSM11]	The Federal `Common Simple Data Elements` schema module MUST be named "`fed:CommonSimpleDataElements Schema Module`"

3.6.3.3 Unqualified Datatypes

[SSM12]	A schema module defining all Federal `Unqualified Datatypes` MUST be created.

[SSM13]	The Federal `Unqualified Datatype` schema module MUST be named "`fed:Unqualified Datatype Schema Module`"

3.6.3.4 Qualified Datatypes

[SSM14]	A schema module defining all Federal `Qualified Datatypes` MUST be created.

[SSM15]	The Federal `Qualified Datatypes` schema module MUST be named "`fed:Qualified Datatypes schema module`"

3.6.4 Department and Agency Modularity Options

Agencies are given a great deal of flexibility in their own schema modularity approaches. They may choose to simply promote all schema artifacts to federal level for inclusion in the appropriate federal schema. They may choose to create single Agency schema modules for complex data elements and simple data elements. Agencies may choose to create a schema module for each individual XSD artifact. They must however ensure that they do not reinvent existing artifacts – rather they should reuse available federal components and tailor those through derivation by extension or restriction.

Agencies must also reuse the datatypes available in the federal unqualified and qualified datatypes schema modules. the federal unqualified and qualified datatype schema modules will serve as the basis of – are imported into – Agency unique unqualified and qualified schema modules.

[SSM16]	Agencies MAY create Agency level schema modules for reusable components not included in Federal level schema. Agencies SHOULD submit all Agency reusable components for consideration as Federal level reusable components.

3.6.4.1 AGENCY COMPLEX DATA ELEMENT SCHEMA MODULES

[SSM17]	A schema module defining Agency `Common Complex Data Elements` MAY be created.

[SSM18]	Agency `Common Complex Data Element` schema modules MUST be named "`<agencyToken>:<AgencyName> CommonComplexDataElements Schema Module`"

3.6.4.2 AGENCY COMMON SIMPLE DATA ELEMENTS

[SSM19]	A schema module defining all Agency `Common Simple Data Elements` MAY be created.

[SSM20]	Agency `Common Simple Data Elements` schema modules MUST be named "`fed: <agencyToken>:<AgencyName> Common Simple Data Elements Schema Module`"

3.6.4.3 UNQUALIFIED DATATYPES

[SSM21]	A schema module defining all Agency Unqualified Datatypes MAY be created.

[SSM22]	Agency Unqualified Datatype schema modules MUST be named "`<agencyToken>:<AgencyName> Unqualified Datatype Schema Module`"

3.6.4.4 QUALIFIED DATATYPES

[SSM23]	A schema module defining all Agency Qualified Datatypes MAY be created.

[SSM24]	Agency Qualified Datatypes schema modules MUST be named `<agencyToken>:<AgencyName> Qualified Datatypes schema module`"

3.6.5 Modularity and Namespaces

3.6.5.1 COMMON COMPLEX DATA ELEMENTS SCHEMA MODULES

[NMS8]	Each Federal and Agency `Common Complex Data Elements` Schema Module MUST reside in its own namespace.

[NMS9]	Each Federal and Agency `Common Complex Data Elements` Schema Module MUST be represented by the token "`CCD[agencyid][majorversion][minorversion]`".

Federal schema modules will be easily distinguishable from Agency schema modules through both namespaces and the addition of agency identifiers [agencyid] to namespace token strings.

3.6.5.2 COMMON SIMPLE DATA ELEMENTS SCHEMA MODULES

[NMS10]	Each Federal and Agency `Common Simple Data Elements` Schema Module MUST reside in its own namespace.

[NMS11]	Each `Common Simple Data Elements` schema module MUST be represented by the token "`csd[agencyid][majorversion][minorversion]`".

3.6.5.3 UNQUALIFIED DATATYPE SCHEMA MODULES

[NMS12]	Each Federal and Agency `Unqualified Datatype` schema module MUST reside in its own namespace.

[NMS13]	Each Federal and Agency `Unqualified Datatype` schema module namespace MUST be represented by the token "`udt[agencyid][majorversion][minorversion]`".

3.6.5.4 QUALIFIED DATATYPE SCHEMA MODULES

[NMS14]	Each Federal and Agency `Qualified Datatypes` schema module MUST reside in its own namespace.

[NMS15]	Each Federal and Agency `Qualified Datatypes` schema module namespace MUST be represented by the token "`qdt[agencyid][majorversion][minorversion]`".

3.6.5.5 CODE AND IDENTIFIER LIST SCHEMA MODULES

[NMS16]	Each `CodeList` schema module MUST be maintained in a separate namespace.

3.7 DOCUMENTATION

3.7.1 Annotation

Care must be given when using the xsd:annotation element. Although the leaf xsd:documentation element discussed in the next section provides no problems, the leaf xsd:appInfo provides significant security risks and must not be used.

[GXS12]	Federal or Agency schema MUST NOT use `xsd:appinfo`.

3.7.2 Embedded Documentation

[GXS2]	Federal and Agency schema should provide two normative schemas for each transaction. One schema shall be fully annotated. One schema shall be a run-time schema devoid of documentation.

[DOC1]	The `xsd:documentation` element for every Datatype MUST contain a structured set of annotations in the following sequence and pattern:

- ComponentType (mandatory): The type of component to which the object belongs. For Datatypes this must be "DT".
- DictionaryEntryName (mandatory): The official name of a Datatype.
- Version (optional): An indication of the evolution over time of the Datatype.
- Definition(mandatory): The semantic meaning of a Datatype.
- ObjectClassQualifier (optional): The qualifier for the object class.
- ObjectClass(optional): The Object Class represented by the Datatype.
- RepresentationTerm (mandatory): A Representation Term is an element of the name which describes the form in which the property is represented.
- DataTypeQualifier (optional): semantically meaningful name that differentiates the Datatype from its underlying Core Component Type.
- DataType (optional): Defines the underlying Core Component Type.

[DOC2]	A Datatype definition MAY contain one or more Content Component Restrictions to provide additional information on the relationship between the Datatype and its corresponding Core Component Type. 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 Qualified Datatype definition MAY contain one or more allowed metadata attribute restrictions to provide additional information on the relationship between the Datatype and its corresponding unqualified Datatype. If used the metadata Restrictions must contain a structured set of annotations in the following patterns:

- MetadataAttributeName (mandatory): Identifies the metadata attribute on which the restriction applies.
- RestrictionValue (mandatory, repetitive): The actual value(s) that is (are) valid for the metadata attribute

[DOC4]	The `xsd:documentation` element for every simple data element MUST contain a structured set of annotations in the following sequence and pattern:

- DictionaryEntryName (mandatory): The ISO 11179 conformant name.
- Version (optional): An indication of the evolution over time of the simple data element.
- Definition(mandatory): The semantic meaning of the simple data element.
- Cardinality(mandatory): Indication whether the simple data element represents a not-applicable, optional, mandatory and/or repetitive characteristic of higher level aggregates.
- ObjectClassQualifier (optional): The qualifier for the object class.
- ObjectClass(mandatory): The Object Class of which the simple data element is a property of.
- PropertyTermQualifier (optional): The qualifier for the property term.
- PropertyTerm(mandatory): Property Term represents the distinguishing characteristic or Property of the Object Class and shall occur naturally in the definition of the simple data element.
- RepresentationTerm (mandatory): A Representation Term describes the form in which the simple data element is represented.
- DataTypeQualifier (optional): semantically meaningful name that differentiates the Qualified Datatype of the simple data element from its underlying Unqualified Datatype.

- DataType (mandatory): Defines the Datatype used for the simple data element.
- AlternativeBusinessTerms (optional): Any synonym terms under which the Simple data element is commonly known and used in the business.
- Examples (optional): Examples of possible values for the Simple data element.

[DOC5] The `xsd:documentation` element for every complex data element representing a class MUST contain a structured set of annotations in the following sequence and pattern:

- ComponentType (mandatory): The type of component to which the object belongs. For classes, this must be “complex data element”.
- DictionaryEntryName (mandatory): The official name of the complex data element.
- Version (optional): An indication of the evolution over time of the complex data element.
- Definition(mandatory): The semantic meaning of the complex data element.
- ObjectClassQualifier (optional): The qualifier for the object class.
- ObjectClass(mandatory): The Object Class represented by the complex data element.
- AlternativeBusinessTerms (optional): Any synonym terms under which the Complex data element is commonly known and used in the business.

[DOC6] The `xsd:documentation` element for every Association Property element declaration MUST contain a structured set of annotations in the following sequence and pattern:

- AssociationType (mandatory): The nature of the association to which the object belongs.
- DictionaryEntryName (mandatory): The official name of the Association Property.
- Version (optional): An indication of the evolution over time of the Association Property.
- Definition(mandatory): The semantic meaning of the Association Property.
- Cardinality(mandatory): Indication whether the Association Property represents an optional, mandatory and/or repetitive assocation.

- ObjectClass(mandatory): The Object Class containing the Association Property.
- PropertyTermQualifier (optional): A qualifier is a word or words which help define and differentiate the Association Property.
- PropertyTerm(mandatory): The nature of the association between the two complex data elements.
- AssociatedObjectClassQualifier (optional): Associated Object Class Qualifiers describe the 'context' of the relationship with another complex data element.
- AssociatedObjectClass (mandatory); Associated Object Class is the Object Class at the other end of this association. It represents the child Class contained by the property of the parent class.

3.7.3 Schema Guides

4 Naming XML Constructs

4.1 General Naming Rules

4.1.1 Syntax Neutral Naming Rules

A Naming Convention is necessary to gain consistency in the naming and defining of all federal data elements. The resulting consistency facilitates comparison during the discovery and analysis process, and precludes ambiguity, such as the development of multiple complex data elements with different names that have the same semantic meaning.

The Naming Convention is derived from the guidelines and principles described in document ISO 11179 Part 5 -- Naming and Identification Principles For Data Elements. In certain instances, these guidelines have been adapted to the XML environment.

All official dictionary entries will be in English. Due to the growing exchange of data between federal agencies and coalition partners, an authoritative source that will ensure absolute clarity and understanding of the names and definitions is required. The Oxford English Dictionary is that authoritative source. Specifically, Oxford English Dictionary American spellings will be used as the primary source.

A supplementary Controlled Vocabulary of Object Classes, Property Terms, and Qualifiers will be developed to identify the definition to be used for any words that are potentially ambiguous. This Controlled Vocabulary shall also be used to identify the preferred word in cases where more than one word might be used to cover the same definition. The Controlled Vocabulary will also contain terms not found in the Oxford English Dictionary. This will ensure that each word within any of the names and definitions is used in a consistent and unambiguous way. The resultant semantic integrity will also mean that translation into other languages retains the precise original meaning.

4.1.1.1 Dictionary Information

Each *data element* contains the following dictionary information that is impacted by the naming rules in subsequent sub-sections:

- ***Dictionary Entry Name*** (Mandatory). This is the unique official name of the *Data Element* in the dictionary.

- ***Definition*** (Mandatory). This is the unique *Business Semantic* of that *Data Element*.

- ***Business Term*** (Optional). This is a synonym term under which the *data Element i*s commonly known and used in the business. A *Data Element* may have several *Business Terms* or synonyms.

[Example]

Dictionary Entry Name – **`Person. Tax. Identifier`**

Definition – The registered national tax identification of a person

Business Term – Income tax number, national register number, personal tax register number, social security number, national insurance number

The naming rules are also based on the following concepts as defined in ISO 11179:

- ***Object Class***. This represents the logical data grouping or aggregation (in a logical data model) to which a *Property* belongs. The *Object Class* is expressed by an *Object Class Term*. The *Object Class* thus is the part of a *Dictionary Entry Name* of the *Data Element* that represents an activity or object in a specific *Context*. *Object Classes* have explicit boundaries and meaning and their *Properties* and behaviour follow the same rules.

- ***Property Term***. This represents the distinguishing characteristic or *Property* of the *Object Class* and shall occur naturally in the definition.

- ***Representation Term***. An element of the *Data Element* name which describes the form in which the *Data Element* is represented.

- ***Qualifier Term***. A word or words which help define and differentiate a *Data Element* from its other related *Data Elements*.

4.1.1.2 GENERAL RULES

[DEN1]	The dictionary content, with the exception of *Business Terms*, shall be in the *English Language* following the primary *Oxford English Dictionary* American spellings to assure unambiguous spelling.

4.1.1.3 RULES FOR DEFINITIONS

[DEN2]	The definition shall be consistent with the requirements of ISO 11179-4 Section 4 and will provide an understandable meaning, which should also be translatable to other languages.

[DEN3]	The definition shall take into account the fact that the users of the Data Elements are not necessarily native English speakers. It shall therefore contain short sentences, using normal words. Wherever synonym terms are possible, the definition shall use the preferred term as identified in the Controlled Vocabulary.

[DEN4]	The definition of a Simple Data Element shall use a structure that is based on the existence of the Object Class Term, the Property Term, the Data Type, and any Qualifiers.

[DEN5]	The definition of an Association between Complex Data Elements shall use a structure that is based on the existence of the Object Class Term of the associating Complex Data Element, the Property (nature of the association), and the Object Class Term of the associated Complex Data Element and any Qualifiers.

[DEN6]	Whenever both the definite (i.e. **the**) and indefinite article (i.e. **a**) are possible in a definition, preference shall be given to an indefinite article (i.e. **a**).

[Note]

To verify the quality of the definition, place the *Dictionary Entry Name* followed by *is* before the definition to ensure that it is not simply a repetition of the *Dictionary Entry Name*.

4.1.1.4 RULES FOR DICTIONARY ENTRY NAMES

[DEN7]	The *Dictionary Entry Name* shall be unique.

[DEN8]	The *Dictionary Entry Name* shall be extracted from the definition.

[DEN9]	The *Dictionary Entry Name* shall be concise and shall not contain consecutive redundant words.

[DEN10]	The *Dictionary Entry Name* and all its components shall be in singular form unless the concept itself is plural.

[Example]

The singular **`Good`** does not exist, whereas **`Goods`** is a plural noun whose concept involves one or multiple (plural) items

[DEN11]	The *Dictionary Entry Name* shall not use non-alphanumeric characters unless required by language rules. Numeric characters should not be used for sequencing.

[DEN12]	The *Dictionary Entry Name* shall only contain verbs, nouns and adjectives (i.e. no words like *and, of, the*, etc.).

[DEN13]	Abbreviations and acronyms that are part of the *Dictionary Entry Name* shall be expanded or explained in the definition.

[DEN14]	The *Object Class Term, Property Term*, and *Representation Term* components of a *Dictionary Entry Name* shall be separated by dots. The space character shall separate words in multi-word *Object Class Terms* and/or multiword *Property Terms*, including their *Qualifier Terms*. Every word shall start with a capital letter. *Qualifier Terms* shall be separated from their associated *Object Class* or *Property Term* by an

underscore (_) followed by a space to separate each qualifier. To allow spell checking of the words in the *Dictionary Entry Nam*e, a space character shall follow the dots after *Object Class Term(s)* and *Property Term(s)*.

[DEN15] *Qualifier Terms* shall precede the associated *Object Class Term* or *Property Term.* The order of qualifiers shall not be used to differentiate *Dictionary Entry Names*.

[DEN16] The *Dictionary Entry Name* of a Simple Data Element shall consist of the following parts in the order specified:

the Object Class Term of the owning the corresponding Basic Core Component Property,

the *Property Term* of the corresponding class property, and

the Representation Term of the Data Type

any Qualfiying Terms

[Example]

Tax. Description. Text

[DEN17] The *Dictionary Entry Name* of an Complex Data Element *Association* shall consist of the following components in the specified order:

the Object Class Term of the Complex Data Element owning the corresponding Association Property,

the Property Term of the corresponding Association Property,

the Object Class Term of the Complex Data Element on which the corresponding Association Core Component Property is based, and

Any Qualifying Terms.

[Example]

Person. Residence. Address

[DEN18] The components of a *Dictionary Entry Name* shall be separated by dots. The space character shall separate words in multi-word *Object Class Terms* and/or multi-word *Property Terms*. Every word shall start with a capital letter. To allow spell checking of the *Directory Entry Names*' words, the dots after *Object Class Terms* and *Property Terms* shall be followed by a space character.

[Note]

The use of CamelCase for *Dictionary Entry Names* has been considered, but has been rejected for following reasons:

- Use of CamelCase will not allow the use of spell checkers
- Strict use of CamelCase makes it impossible to use separators (".") and therefore doesn't allow an unambiguous identification of the composing parts of the *Dictionary Entry Name.*

[DEN19] The name of an *Object Class* shall always have the same semantic meaning throughout the dictionary and may consist of more than one word.

[DEN20] The name of a *Property Term* shall occur naturally in the definition and may consist of more than one word. A name of a *Property Term* shall be unique within the *Context* of an *Object Class* but may be reused across different *Object Classes*.

[Example]

Car. Color. Code and Shirt. Color. Code may both exist.

[DEN21] The *Dictionary Entry Name* of an *Complex Data Type* shall consist of a meaningful *Object Class Term*. The *Object Class Term* may consist of more than one word.

[Example]

Postal Address; Party

4.1.2 XML Naming Rules

[GNR1]	XML element, attribute and type names MUST be in the English language, using the primary American spellings provided in the Oxford English Dictionary for writers and editors.

[GNR2]	XML element, attribute and type names MUST be consistently derived from ISO 11179 conformant dictionary entry names.

[GNR3]	XML element, attribute and type names constructed from dictionary entry names MUST NOT include periods, spaces, other separators, or characters not allowed by W3C XML 1.0 for XML names.

[GNR4]	XML element, attribute, and simple and complex type names MUST NOT use acronyms, abbreviations, or other word truncations, except those in the list of exceptions published in Appendix XX.

[GNR5]	Acronyms and abbreviations MUST only be added to the federal approved acronym and abbreviation list after careful consideration for maximum understanding and reuse.

[GNR6]	The acronyms and abbreviations listed in Appendix XX MUST always be used.

[GNR7]	XML element, attribute and type names MUST be in singular form unless the concept itself is plural.

[GNR8]	The UpperCamelCase (UCC) convention MUST be used for naming elements and types.

[GNR9]	The lowerCamelCase (LCC) convention MUST be used for naming attributes.

4.2 TYPE NAMING RULES

4.2.1 Complex Type Names for Complex Data Elements

[CTN1]	An `xsd:complexType` name based on a complex data element MUST be the `Dictionary Entry Name` with the separators removed and with with the suffix "`Type`" appended following the upper camel case convention.

4.2.2 Complex Type Names for Simple Data Elements

[CTN2]	An xsd:complexType name based on a Simple Data Element MUST be the Simple Data Element Dictionary Entry Name with the separators removed and with the "Type" suffix appended after the representation term.

4.2.3 Type Names for Unqualified Datatypes

4.2.3.1 COMPLEX TYPE NAMES

[CTN3]	An `xsd:complexType` for a `unqualified datatype` MUST have the name of the corresponding `ccts:CoreComponentType`, with the separators removed and with the "`Type`" suffix appended.

4.2.3.2 SIMPLE TYPE NAMES

[STN1]	Each `xsd:simpleType` definition name MUST be the `datatype` dictionary entry name with the separators removed.

4.2.4 Type Names for Qualified Datatypes

4.2.4.1 Complex Type Names

4.2.4.2 Simple Type Names

4.2.5 Type Names for Codes and Identifiers

4.2.5.1 Complex Type Names

4.2.5.2 Simple Type Names

4.3 Element Naming Rules

4.3.1 Element Names for Complex Data Elements

4.3.1.1 Element Names for Simple Data Elements

4.3.1.2 Element Names for Associations

4.4 Attribute Naming Rules

[ATN1]	Each `xsd:attribute` "`name`" MUST be an ISO 11179 conformant dictionary entry name object class, property term and representation term with any separators removed.

5 Declarations and Definitions

5.1 TYPE DEFINITIONS

5.1.1 General Type Definitions

[GTD1]	All types MUST be named.

5.1.2 Simple Type Definitions

[STD1]	For every datatype whose metadata components map directly onto the properties of a built-in xsd:DataType, the datatype MUST be defined as a named xsd:simpleType in the fed:unqualifiedDatatype schema module.

5.1.3 Complex Type Definitions

[CTD1]	For every complex and simple data element identified in a Federal or Agency data centric process model, a named `xsd:complexType` MUST be defined.

[CTD2]	Every `complex data element xsd:complexType` definition content model for data-centric schema MUST use the `xsd:sequence` element with appropriate global element references to reflect each simple data element (property) that constitute its content model.

[CTD3]	Every `simple data element xsd:complexType` definition content model MUST use the `xsd:simpleContent` element.

[CTD4]	Every `simple data element` `xsd:complexType` content model `xsd:simpleContent` element MUST consist of an `xsd:extension` element.

[CTD5]	Every `simple data element` `xsd:complexType` content model `xsd:base` attribute value MUST be an unqualified or qualified federal datatype as appropriate.

[CTD6]	For every datatype used in a data model, a named `xsd:complexType` or `xsd:simpleType` MUST be defined.

[CTD7]	All defined or used unqualified datatypes MUST either be those defined in the Unqualified Datatype Schema Module from UN/CEFACT, the federal Unqualified Datatype Schema Module, or an Agency Unqualified Datatype Schema Module.

[CTD8]	Each unqualified Datatype `xsd:complexType` must be based on its corresponding source `xsd:complexType`. Note: If we don't use CTD7, then we need CTD8. If we use CTD7, then we don't need CTD8.

[CTD9]	Every qualified Datatype whose corresponding unqualified datatype is defined as an `xsd:complexType` MUST also be defined as an `xsd:complexType` and MUST be based on the same `xsd:simpleType`.

[CTD10]	Every qualified Datatype whose corresponding `unqualified datatype` is defined as an `xsd:simpleType` MUST also be defined as an `xsd:simpleType` and MUST be based on the same `xsd:simpleType`.

[CTD11]	Each unqualified Datatype `xsd:complexType` definition must contain one `xsd:simpleContent` element.

[CTD12]	For every `datatype` whose metadata components are not equivalent to the properties of a built-in `xsd:Datatype`, it MUST be defined as a named `xsd:complexType` in the Unqualified Datatype schema module.

[CTD13]	The `Unqualified Datatype xsd:complexType` definition `xsd:simpleContent` element MUST contain one `xsd:extension` element. This `xsd:extension` element MUST include an `xsd:base` attribute that defines the specific `xsd:Built-inDatatype` required.

[CTD14]	Each `metadata component xsd:attribute` "type" MUST define the specific `xsd:Built-in Datatype` or the user defined `xsd:simpleType` for the `metadata component of the unqualified Datatype`.

[CTD15]	Each `metadata component xsd:attribute` "`use`" MUST define the occurrence of that `metadata component` as either "`required`", or "`optional`".

5.1.3.1 CONTENT MODELS

5.1.3.1.1 Sequence

5.1.3.1.2 xsd:all

[GXS8]	The `xsd:all` element MUST NOT be used in data centric schema.

5.1.3.1.3 xsd:choice

[GXS9]	The `xsd:choice` element SHOULD NOT be used where customisation and extensibility are a concern.

5.1.3.2 XSD:INCLUDE

[GXS10]	The xsd:include feature MUST only be used within a root schema.

5.1.3.3 XSD:UNION

[GXS11]	The `xsd:union` technique MUST NOT be used except for Code and Identifier Lists. The `xsd:union` technique MAY be used for Code and Identifier Lists.

5.1.3.4 XSD:ANYTYPE

[GTD2]	The `xsd:anyType` MUST NOT be used.

5.2 ELEMENT DECLARATIONS

5.2.1 Global and Local Elements

5.2.2 Elements Bound to Complex Types

[ELD3]	For every complex data element identified in the data model, a global element bound to the corresponding `xsd:complexType` MUST be declared.

5.2.3 Elements Bound to Simple Types

5.2.4 Elements Representing Associations between Complex Data Elements

[ELD4]	If an association between two complex data elements is unqualified, the association MUST use the global element declared for the associated complex data element. If an association between two complex data elements is qualified, a new global element representing the qualified association MUST be declared and used.

5.2.5 Elements Bound to Datatypes

[ELD5]	For each `datatype SimpleType definition`, an xsd:restriction element MUST be declared.

5.2.6 Elements representing Code Lists and Identifier Lists

[ELD6]	Code list and Identifier List `xsd:import` elements MUST contain the namespace and schema location attributes.

5.2.7 Empty Elements

[ELD7]	Empty elements MUST not be declared.

5.2.8 XSD:Any Elements

[ELD8]	The `xsd:any` element MUST NOT be used.

5.3 ATTRIBUTE DECLARATIONS

5.3.1 User Defined Attributes

[ATD1]	User defined attributes SHOULD NOT be used in data centric schema. When used, user defined attributes MUST only convey supplemental metadata information from table XX not intended for storage or application processing.

5.3.2 Metadata Attributes

A general discussion here on the role of metadata attributes and their importance. Explain how metadata attributes convey data that is not normally stored. Identify the table in the back that will convey the approved list of metadata attributes (taken from table 8-2 in ISO 15000-5).

5.3.3 Global Attributes

[ATD2]	If a Schema Expression contains one or more common attributes that apply to all elements contained or included or imported therein, the common attributes MUST be declared as part of a global attribute group.

5.3.4 Using Attribute Groups

5.3.5 Schema Location

[ATD3]	Each `xsd:schemaLocation` attribute declaration MUST contain a system-resolvable URL. This system resolvable URL SHOULD be persistent and accessable from any query. If the system resolvable URL is not made publicly available, it SHOULD be a relative URL referencing the location of the schema or schema module in the release package.

5.3.6 XSD:Nil

[ATD4]	The xsd built in nillable attribute MUST NOT be used for data centric schema that require authentication and nonrepudiation

5.3.7 XSD:anyAttribute

[ATD5]	The `xsd:anyAttribute` MUST NOT be used for data centric schema. The xsd:anyAttribute MAY be used for document-centric schema if consistency in not an issue.

6 Extending and Restricting Types

[GXS13]	Complex Type extension or restriction MAY be used where appropriate.

6.1 GUIDELINES FOR EXTENSION

6.2 GUIDELINES FOR RESTRICTION

6.3 XSD:SUBSTITUTIONGROUP

[GXS5]	The `xsd:SubstitutionGroups` feature Should NOT be used In data centric schema. If used, it should only be used in user defined customization schema or when extending Agency or Federal XSD components.

6.4 XSD:FINAL

[GXS6]	The `xsd:final` attribute SHOULD be used where appropriate to control undesireable extensions.

7 Code Lists and Identifier Lists

[CIL1]	All Codes and Identifiers MUST be part of an Agency or externally maintained Code List.

[CIL2]	Agency Libraries MUST identify and use external standardized code and Identifier lists when available rather than develop their own native code lists.

[CIL3]	Agency Libraries MAY design and use an internal code or identifier list where an existing external code or identifier list needs to be extended, or where no suitable external code or identifier list exists.

[CIL4]	All Federal or Agency maintained or used Code and Identifier Lists MUST be enumerated using the Federal Code and Identifier List Schema Module Template.

[CIL5]	The name of each Federal or Agency Code or Identifier List Schema Module MUST be of the form: `{Owning Organization}{Code or Identifier List Name}{Code List Schema Module}`

[CIL6]	An `xsd:import` element MUST be declared for every code or identifier list required in a root schema.

[CIL7]	Users of the Federal or Agency Library MAY identify any subset they wish from an identified code or identifier list for their own trading community conformance requirements.

[CIL8]	The `xsd:schemaLocation` declaration for code and identifier list schema modules MUST include the complete URI used to identify the relevant schema.

8 Using Schematron

9 Miscellaneous XSD Rules

9.1 XSD:SIMPLETYPE

[GXS3]	Built-in xsd:simpleType SHOULD be used wherever possible.

9.2 NAMESPACE DECLARATION

[GXS4]	All W3C XML Schema constructs in federal and Agency Schema and schema modules MUST contain the following namespace declaration on the xsd schema element: xmlns:xsd="http://www.w3.org/2001/XMLSchema"

9.3 XSD:NOTATION

[GXS7]	`xsd:notations` MUST NOT be used.

9.4 XSD:APPINFO

[GXS12]	Federal or Agency schema MUST NOT use `xsd:appinfo`.

9.5 XSD:KEY AND XSD:KEYREF

10 XML Instances

10.1 VALIDATION

An XML instance is well-formed if it conforms to XML 1.0 It is valid if it conforms to a supporting DTD or Schema. To ensure consistency, federal XML must necessarily be well formed and valid. As XSD is defined as the normative form for XML schema expressions for the federal government, by default, every federal XML instance must have a supporting fully conformant XSD Schema.

[IND1]	All instance documents MUST validate to a corresponding XSD schema.

10.2 CHARACTER ENCODING

XML Instances can be expressed in several different character encodings. To ensure consistency, Federal XML instances should clearly identify the character encoding being used.

[IND2]	Instance documents MUST always identify their character encoding with the XML declaration

At the International level, consensus has been reached between the four de jure standards bodies as the preferred character set encoding for XML instance documents to facilitate interoperability. This consensus is expressed in ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83) which calls for the use of UTF-8. Federal XML instances should follow these guidelines.

[IND3]	In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83), all federal or Agency XML SHOULD be expressed using UTF-8.

10.3 ROOT ELEMENT

[RED1]	Every instance document must use the global element defined as the root element in the schema as its root element.

10.4 XML SCHEMA INSTANCE ATTRIBUTE NAMESPACE DECLARATION

The XML Schema Specification provides for attributes that are commonly used in instance documents but that do not appear in a schema. These attributes belong to a specific namespace. To ensure consistency, this namespace will be declared in each Federal XML instance.

[IND4]	All instance documents MUST contain the following namespace declaration in the root element: `xmlns:xsi`="http://www.w3.org/2001/XMLSchema-instance

10.5 EMPTY ELEMENTS AND NULL VALUES

Empty elements and null values, although of some value in databases, is extremely dangerous in data centric information exchanges as it significantly impacts on the trustworthyness of aninstance document.

[IND5]	Data centric instance documents MUST NOT contain an element devoid of content or null values.

Conversely, schema implementation guides must not indicate that if data is not present, a particular meaning is conveyed.

[IND6]	The absence of a construct or data in an instance document MUST NOT carry meaning.

Appendix A. Federal XML Naming and Design Rules Checklist

This Appendix contains a checklist of all Fedral XML Naming and Design rules in rule category alphabetical order as follows:

- Attribute Declaration Rules (ATD)
- Attribute Naming Rules (ATN)
- Code and Identifier List Rules (CIL)
- ComplexType Definition Rules (CTD)
- ComplexType Naming Rules (CTN)
- Data Element Dictionary Entry Names and Definitions (DEN)
- Documentation Rules (DOC)
- Element Declaration Rules (ELD)
- General Naming Rules (GNR)
- General Type Definition Rules (GTD)
- General XML Schema Rules (GXS)
- Instance Document Rules (IND)
- Modeling Constraints Rules (MDC)
- Naming Constraints Rules (NMC)
- Namespace Rules (NMS)
- Root Element Declaration Rules (RED)
- Schema Structure Modularity Rules (SSM)

- Standards Adherence Rules (STA)
- SimpleType Naming Rules (STN)
- SimpleType Definition Rules (STD)
- Versioning Rules (VER)

A.1 Attribute Declaration Rules

[ATD1]	User defined attributes SHOULD NOT be used in data centric schema. When used, user defined attributes MUST only convey supplemental metadata information from table XX not intended for storage or application processing.
[ATD2]	If a Schema Expression contains one or more common attributes that apply to all elements contained or included or imported therein, the common attributes MUST be declared as part of a global attribute group.
[ATD3]	Each `xsd:schemaLocation` attribute declaration MUST contain a system-resolvable URL. This system resolvable URL SHOULD be persistent and accessable from any query. If the system resolvable URL is not made publicly available, it SHOULD be a relative URL referencing the location of the schema or schema module in the release package.
[ATD4]	The xsd built in nillable attribute MUST NOT be used for data centric schema that require authentication and nonrepudiation
[ATD5]	The `xsd:anyAttribute` MUST NOT be used for data centric schema. The xsd:anyAttribute MAY be used for document-centric schema if consistency in not an issue.

A.2 Attribute Naming Rules

[ATN1]	Each `xsd:attribute` "`name`" MUST be an ISO 11179 conformant dictionary entry name object class, property term and representation term with any separators removed.

A.3 Code and Identifier List Rules

[CIL1]	All Codes and Identifiers MUST be part of an Agency or externally maintained Code List.
[CIL2]	Agency Libraries MUST identify and use external standardized code and Identifier lists when available rather than develop their own native code lists.
[CIL3]	Agency Libraries MAY design and use an internal code or identifier list where an existing external code or identifier list needs to be extended, or where no suitable external code or identifier list exists.
[CIL4]	All Federal or Agency maintained or used Code and Identifier Lists MUST be enumerated using the Federal Code and Identifier List Schema Module Template.
[CIL5]	The name of each Federal or Agency Code or Identifier List Schema Module MUST be of the form: `{Owning Organization}{Code or Identifier List Name}{Code List Schema Module}`
[CIL6]	An `xsd:import` element MUST be declared for every code or identifier list required in a root schema.
[CIL7]	Users of the Federal or Agency Library MAY identify any subset they wish from an identified code or identifier list for their own trading community conformance requirements.
[CIL8]	The `xsd:schemaLocation` declaration for code and identifier list schema modules MUST include the complete URI used to identify the relevant schema.

A.4 ComplexType Definition Rules

[CTD1]	For every complex and simple data element identified in a Federal or Agency data centric process model, a named `xsd:complexType` MUST be defined.
[CTD2]	Every `complex data element xsd:complexType` definition content model for data-centric schema MUST use the `xsd:sequence` element with appropriate global element references to reflect each simple data element (property) that constitute its content model.
[CTD3]	Every `simple data element xsd:complexType` definition content model MUST use the `xsd:simpleContent` element.
[CTD4]	Every `simple data element xsd:complexType` content model `xsd:simpleContent` element MUST consist of an `xsd:extension` element.
[CTD5]	Every `simple data element xsd:complexType` content model `xsd:base` attribute value MUST be an unqualified or qualified federal datatype as appropriate.
[CTD6]	For every datatype used in a data model, a named `xsd:complexType` or `xsd:simpleType` MUST be defined.
[CTD7]	All defined or used unqualified datatypes MUST either be those defined in the Unqualified Datatype Schema Module from UN/CEFACT, the federal Unqualified Datatype Schema Module, or an Agency Unqualified Datatype Schema Module.
[CTD8]	Each unqualified Datatype `xsd:complexType` must be based on its corresponding source `xsd:complexType`. Note: If we don't use CTD7, then we need CTD8. If we use CTD7, then we don't need CTD8.

A.4 ComplexType Definition Rules

[CTD9]	Every qualified Datatype whose corresponding unqualified datatype is defined as an `xsd:complexType` MUST also be defined as an `xsd:complexType` and MUST be based on the same `xsd:simpleType`.
[CTD10]	Every qualified Datatype whose corresponding `unqualified datatype` is defined as an `xsd:simpleType` MUST also be defined as an `xsd:simpleType` and MUST be based on the same `xsd:simpleType`.
[CTD11]	Each unqualified Datatype `xsd:complexType` definition must contain one `xsd:simpleContent` element.
[CTD12]	For every `datatype` whose metadata components are not equivalent to the properties of a built-in `xsd:Datatype`, it MUST be defined as a named `xsd:complexType` in the Unqualified Datatype schema module.
[CTD13]	The `Unqualified Datatype xsd:complexType` definition `xsd:simpleContent` element MUST contain one `xsd:extension` element. This `xsd:extension` element MUST include an `xsd:base` attribute that defines the specific `xsd:Built-inDatatype` required.
[CTD14]	Each `metadata component xsd:attribute` "type" MUST define the specific `xsd:Built-in Datatype` or the user defined `xsd:simpleType` for the `metadata component of the unqualified Datatype`.
[CTD15]	Each `metadata component xsd:attribute` "`use`" MUST define the occurrence of that `metadata component` as either "`required`", or "`optional`".

A.5 ComplexType Naming Rules

[CTN1]	An `xsd:complexType` name based on a `class` MUST be the `Dictionary Entry Name` with the separators removed and with with the suffix "`Type`" appended following the upper camel case convention.
[CTN2]	An `xsd:complexType` name based on a Simple Data Element MUST be the `Simple Data Element Dictionary Entry Name` with the separators removed and with the "`Type`" suffix appended after the representation term.???
[CTN3]	An `xsd:complexType` for a `unqualified datatype` MUST have the name of the corresponding `ccts:CoreComponentType`, with the separators removed and with the "`Type`" suffix appended.

A.6 Data Element Dictionary Entry Names and Definitions

[DEN1]	The dictionary content, with the exception of *Business Terms*, shall be in the *English Language* following the primary *Oxford English Dictionary* American spellings to assure unambiguous spelling.
[DEN2]	The definition shall be consistent with the requirements of ISO 11179-4 Section 4 and will provide an understandable meaning, which should also be translatable to other languages.
[DEN3]	The definition shall take into account the fact that the users of the Data Elements are not necessarily native English speakers. It shall therefore contain short sentences, using normal words. Wherever synonym terms are possible, the definition shall use the preferred term as identified in the Controlled Vocabulary.
[DEN4]	The definition of a Simple Data Element shall use a structure that is based on the existence of the Object Class Term, the Property Term, the Data Type, and any Qualifiers.

[DEN5]	The definition of an Association between Complex Data Elements shall use a structure that is based on the existence of the Object Class Term of the associating Complex Data Element, the Property (nature of the association), and the Object Class Term of the associated Complex Data Element and any Qualifiers.
[DEN6]	Whenever both the definite (i.e. **the**) and indefinite article (i.e. **a**) are possible in a definition, preference shall be given to an indefinite article (i.e. **a**).
[DEN7]	The *Dictionary Entry Name* shall be unique.
[DEN8]	The *Dictionary Entry Name* shall be extracted from the definition.
[DEN9]	The *Dictionary Entry Name* shall be concise and shall not contain consecutive redundant words.
[DEN10]	The *Dictionary Entry Name* and all its components shall be in singular form unless the concept itself is plural.
[DEN11]	The *Dictionary Entry Name* shall not use non-alphanumeric characters unless required by language rules. Numeric characters should not be used for sequencing.
[DEN12]	The *Dictionary Entry Name* shall only contain verbs, nouns and adjectives (i.e. no words like *and, of, the*, etc.).
[DEN13]	Abbreviations and acronyms that are part of the *Dictionary Entry Name* shall be expanded or explained in the definition.
[DEN14]	The *Object Class Term, Property Term,* and *Representation Term* components of a *Dictionary Entry Name* shall be separated by dots. The space character shall separate words in multi-word *Object Class Terms* and/or multiword *Property Terms,* including their *Qualifier Terms*. Every word shall start with a capital letter. *Qualifier Terms* shall be separated from their associated *Object Class* or *Property Term* by an underscore (_) followed by a space to separate each qualifier. To allow spell checking of the words in the *Dictionary Entry Nam*e, a space character shall follow the dots after *Object Class Term(s)* and *Property Term(s).*

[DEN15]	*Qualifier Terms* shall precede the associated *Object Class Term* or *Property Term.* The order of qualifiers shall not be used to differentiate *Dictionary Entry Names*.
[DEN16]	The *Dictionary Entry Name* of a Simple Data Element shall consist of the following parts in the order specified: the Object Class Term of the owning the corresponding Basic Core Component Property, the *Property Term* of the corresponding class property, and the Representation Term of the Data Type any Qualfiying Terms
[DEN17]	The *Dictionary Entry Name* of an Complex Data Element *Association* shall consist of the following components in the specified order:the Object Class Term of the Complex Data Element owning the corresponding Association Property,the Property Term of the corresponding Association Property, the Object Class Term of the Complex Data Element on which the corresponding Association Core Component Property is based, and Any Qualifying Terms.
[DEN18]	The components of a *Dictionary Entry Name* shall be separated by dots. The space character shall separate words in multi-word *Object Class Terms* and/or multi-word *Property Terms*. Every word shall start with a capital letter. To allow spell checking of the *Directory Entry Names*' words, the dots after *Object Class Terms* and *Property Terms* shall be followed by a space character.
[DEN19]	The name of an *Object Class* shall always have the same semantic meaning throughout the dictionary and may consist of more than one word.
[DEN20]	The name of a *Property Term* shall occur naturally in the definition and may consist of more than one word. A name of a *Property Term* shall be unique within the *Context* of an *Object Class* but may be reused across different *Object Classes*.
[DEN21]	The *Dictionary Entry Name* of an *Complex Data Type* shall consist of a meaningful *Object Class Term*. The *Object Class Term* may consist of more than one word.

A.7 Documentation Rules

[DOC1]	The `xsd:documentation` element for every Datatype MUST contain a structured set of annotations in the following sequence and pattern: ◆ ComponentType (mandatory): The type of component to which the object belongs. For Datatypes this must be "DT". ◆ DictionaryEntryName (mandatory): The official name of a Datatype. ◆ Version (optional): An indication of the evolution over time of the Datatype. ◆ Definition(mandatory): The semantic meaning of a Datatype. ◆ ObjectClassQualifier (optional): The qualifier for the object class. ◆ ObjectClass(optional): The Object Class represented by the Datatype. ◆ RepresentationTerm (mandatory): A Representation Term is an element of the name which describes the form in which the property is represented. ◆ DataTypeQualifier (optional): semantically meaningful name that differentiates the Datatype from its underlying Core Component Type. ◆ DataType (optional): Defines the underlying Core Component Type.

A.7 Documentation Rules

[DOC2]	A Datatype definition MAY contain one or more Content Component Restrictions to provide additional information on the relationship between the Datatype and its corresponding Core Component Type. 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 Qualified Datatype definition MAY contain one or more allowed metadata attribute restrictions to provide additional information on the relationship between the Datatype and its corresponding unqualified Datatype. If used the metadata Restrictions must contain a structured set of annotations in the following patterns: ◆ MetadataAttributeName (mandatory): Identifies the metadata attribute on which the restriction applies. ◆ RestrictionValue (mandatory, repetitive): The actual value(s) that is (are) valid for the metadata attribute
[DOC4]	The `xsd:documentation` element for every simple data element MUST contain a structured set of annotations in the following sequence and pattern: ◆ DictionaryEntryName (mandatory): The ISO 11179 conformant name. ◆ Version (optional): An indication of the evolution over time of the simple data element. ◆ Definition(mandatory): The semantic meaning of the simple data element.

A.7 Documentation Rules

- Cardinality(mandatory): Indication whether the simple data element represents a not-applicable, optional, mandatory and/or repetitive characteristic of higher level aggregates.
- ObjectClassQualifier (optional): The qualifier for the object class.
- ObjectClass(mandatory): The Object Class of which the simple data element is a property of.
- PropertyTermQualifier (optional): The qualifier for the property term.
- PropertyTerm(mandatory): Property Term represents the distinguishing characteristic or Property of the Object Class and shall occur naturally in the definition of the simple data element.
- RepresentationTerm (mandatory): A Representation Term describes the form in which the simple data element is represented.
- DataTypeQualifier (optional): semantically meaningful name that differentiates the Qualified Datatype of the simple data element from its underlying Unqualified Datatype.
- DataType (mandatory): Defines the Datatype used for the simple data element.
- AlternativeBusinessTerms (optional): Any synonym terms under which the Simple data element is commonly known and used in the business.
- Examples (optional): Examples of possible values for the Simple data element.

A.7 Documentation Rules

[DOC5]	The `xsd:documentation` element for every complex data element representing a class MUST contain a structured set of annotations in the following sequence and pattern: ◆ ComponentType (mandatory): The type of component to which the object belongs. For classes, this must be "complex data element". ◆ DictionaryEntryName (mandatory): The official name of the complex data element. ◆ Version (optional): An indication of the evolution over time of the complex data element. ◆ Definition(mandatory): The semantic meaning of the complex data element. ◆ ObjectClassQualifier (optional): The qualifier for the object class. ◆ ObjectClass(mandatory): The Object Class represented by the complex data element. ◆ AlternativeBusinessTerms (optional): Any synonym terms under which the Complex data element is commonly known and used in the business.

A.7 Documentation Rules

[DOC6]	The `xsd:documentation` element for every Association Property element declaration MUST contain a structured set of annotations in the following sequence and pattern: ◆ AssociationType (mandatory): The nature of the association to which the object belongs. ◆ DictionaryEntryName (mandatory): The official name of the Association Property. ◆ Version (optional): An indication of the evolution over time of the Association Property. ◆ Definition(mandatory): The semantic meaning of the Association Property. ◆ Cardinality(mandatory): Indication whether the Association Property represents an optional, mandatory and/or repetitive assocation. ◆ ObjectClass(mandatory): The Object Class containing the Association Property. ◆ PropertyTermQualifier (optional): A qualifier is a word or words which help define and differentiate the Association Property. ◆ PropertyTerm(mandatory): The nature of the association between the two complex data elements. ◆ AssociatedObjectClassQualifier (optional): Associated Object Class Qualifiers describe the 'context' of the relationship with another complex data element. ◆ AssociatedObjectClass (mandatory); Associated Object Class is the Object Class at the other end of this association. It represents the child Class contained by the property of the parent class.

A.8 Element Declaration Rules	
[ELD1]	Each `Schema` MUST identify one and only one global element declaration that defines the document level container being conveyed 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 "*`This element MUST be conveyed as the root element in any instance document based on this Schema expression.`*"
[ELD2]	All data-centric element declarations MUST be global. All document-centric element declarations SHOULD be global.
[ELD3]	For every complex data element identified in the data model, a global element bound to the corresponding `xsd:complexType` MUST be declared.
[ELD4]	If an association between two complex data elements is unqualified, the association MUST use the global element declared for the associated complex data element. If an association between two complex data elements is qualified, a new global element representing the qualified association MUST be declared and used.
[ELD5]	For each `datatype SimpleType definition`, an xsd:restriction element MUST be declared.
[ELD6]	Code list `xsd:import` elements MUST contain the namespace and schema location attributes.
[ELD7]	Empty elements MUST not be declared.
[ELD8]	The `xsd:any` element MUST NOT be used.

A.9 General Naming Rules

[GNR1]	XML element, attribute and type names MUST be in the English language, using the primary American spellings provided in the Oxford English Dictionary for writers and editors.
[GNR2]	XML element, attribute and type names MUST be consistently derived from ISO 11179 conformant dictionary entry names.
[GNR3]	XML element, attribute and type names constructed from dictionary entry names MUST NOT include periods, spaces, other separators, or characters not allowed by W3C XML 1.0 for XML names.
[GNR4]	XML element, attribute, and simple and complex type names MUST NOT use acronyms, abbreviations, or other word truncations, except those in the list of exceptions published in Appendix XX.
[GNR5]	Acronyms and abbreviations MUST only be added to the federal approved acronym and abbreviation list after careful consideration for maximum understanding and reuse.
[GNR6]	The acronyms and abbreviations listed in Appendix XX MUST always be used.
[GNR7]	XML element, attribute and type names MUST be in singular form unless the concept itself is plural.
[GNR8]	The UpperCamelCase (UCC) convention MUST be used for naming elements and types.
[GNR9]	The lowerCamelCase (LCC) convention MUST be used for naming attributes.

A.10 General Type Definition Rules

[GTD1]	All types MUST be named.
[GTD2]	The `xsd:anyType` MUST NOT be used.

A.11 General XML Schema Rules

[GXS1]	Data-centric Schema MUST conform to the following physical layout as applicable: ◆ XML Declaration ◆ <!-- ===== Copyright Notice ===== --> ◆ Any applicable copyright notice ◆ <!-- ===== xsd:schema Element With Namespaces Declarations ===== --> ◆ xsd:schema element to include version attribute and namespace declarations in the following order: ◆ xmlns:xsd ◆ Target namespace ◆ Default namespace ◆ CommonComplexElements ◆ CommonSimpleElements ◆ Datatypes ◆ Identifier Schemes

A.11 General XML Schema Rules

- Code Lists
- Attribute Declarations – elementFormDefault=”qualified” attributeFormDefault=”unqualified”
- <!-- ===== Imports ===== -->
- CommonComplexElements schema module(s)
- CommonSimpleElements schema module(s)
- Unqualified Datatypes schema module(s)
- Qualified Datatypes schema module(s)
- Code and Identifier List schema module(s)
- <!-- ===== Global Attributes ===== -->
- Global Attributes and Attribute Groups
- <!-- ===== Root Element ===== -->
- Root Element Declaration
- Root Element Type Definition
- <!-- ===== Element Declarations ===== -->
- alphabetized order
- <!-- ===== Type Definitions ===== -->
- All type definitions segregated by simple and complex as follows
- <!-- ===== Complex Data Element Type Definitions ===== -->

A.11 General XML Schema Rules

	◆ alphabetized order of Complex Data Element xsd:complexType definitions ◆ <!-- =====Simple Data Element Type Definitions ===== --> ◆ alphabetized order of simple data element xsd:complexType definitions ◆ <!-- ===== Copyright Notice ===== --> ◆ Required copyright notice.
[GXS2]	Federal and Agency schema should provide two normative schemas for each transaction. One schema shall be fully annotated. One schema shall be a run-time schema devoid of documentation.
[GXS3]	Built-in `xsd:simpleType` SHOULD be used wherever possible.
[GXS4]	All W3C XML Schema constructs in federal and Agency Schema and schema modules MUST contain the following namespace declaration on the xsd schema element: xmlns:xsd="http://www.w3.org/2001/XMLSchema"
[GXS5]	The `xsd:SubstitutionGroups` feature Should NOT be used In data centric schema. If used, it should only be used in user defined customization schema or when extending Agency or Federal XSD components.
[GXS6]	The `xsd:final` attribute SHOULD be used where appropriate to control undesireable extensions.
[GXS7]	`xsd:notations` MUST NOT be used.
[GXS8]	The `xsd:all` element MUST NOT be used in data centric schema.
[GXS9]	The `xsd:choice` element SHOULD NOT be used where customisation and extensibility are a concern.

A.11 General XML Schema Rules

[GXS10]	The `xsd:include` feature MUST only be used within a root schema.
[GXS11]	The `xsd:union` technique MUST NOT be used except for Code and Identifier Lists. The `xsd:union` technique MAY be used for Code and Identifier Lists.
[GXS12]	Federal or Agency schema MUST NOT use `xsd:appinfo`.
[GXS13]	Complex Type extension or restriction MAY be used where appropriate.

A.12 Instance Document Rules

[IND1]	All instance documents MUST validate to a corresponding schema.
[IND2]	Instance documents MUST always identify their character encoding with the XML declaration.
[IND3]	In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83), all federal or Agency XML SHOULD be expressed using UTF-8.
[IND4]	All instance documents MUST contain the following namespace declaration in the root element: `xmlns:xsi`="http://www.w3.org/2001/XMLSchema-instance"
[IND5]	Data centric instance documents MUST NOT contain an element devoid of content or null values.
[IND6]	The absence of a construct or data in an instance document MUST NOT carry meaning.

A.13 Modeling Constraints Rules

[MDC1]	Libraries and Schemas MUST only use approved datatypes.
[MDC2]	Mixed content MUST NOT be used in data centric schema except where contained in an `xsd:documentation` element.

A.14 Naming Constraints Rules

[NMC1]	Each dictionary entry name MUST define one and only one fully qualified path (FQP) for an element or attribute.

A.15 Namespace Rules

[NMS1]	Every schema module, except internal schema modules, MUST have a namespace declared using the `xsd:targetNamespace` attribute.
[NMS2]	Every defined or used schema set version MUST have its own unique namespace
[NMS3]	Federal Namespaces MUST only contain federally developed schema modules
[NMS4]	Agency Namespaces MUST only contain agency developed schema modules
[NMS5]	The namespace names for Schemas holding draft status MUST be of the form: `urn:`
[NMS6]	The namespace names for Schemas holding Approved status MUST be of the form:

A.15 Namespace Rules

	`urn:`
[NMS7]	Published namespaces MUST never be changed.
[NMS8]	Each Federal and Agency `Common Complex Data Elements` Schema Module MUST reside in its own namespace.
[NMS9]	Each Federal and Agency `Common Complex Data Elements` Schema Module MUST be represented by the token "`CCD[agencyid][majorversion][minorversion]`".
[NMS10]	Each Federal and Agency `Common Simple Data Elements` Schema Module MUST reside in its own namespace.
[NMS11]	Each `Common Simple Data Elements` schema module MUST be represented by the token "`csd[agencyid][majorversion][minorversion]`".
[NMS12]	Each Federal and Agency `Unqualified Datatype` schema module MUST reside in its own namespace.
[NMS13]	Each Federal and Agency `Unqualified Datatype` schema module namespace MUST be represented by the token "`udt[agencyid][majorversion][minorversion]`".
[NMS14]	Each Federal and Agency `Qualified Datatypes` schema module MUST reside in its own namespace.
[NMS15]	Each Federal and Agency `Qualified Datatypes` schema module namespace MUST be represented by the token "`qdt[agencyid][majorversion][minorversion]`".
[NMS16]	Each `CodeList` schema module MUST be maintained in a separate namespace.

A.16 Root Element Declaration Rules

[RED1]	Every instance document must use the global element defined as the root element in the schema as its root element.

A.17 Schema Structure Modularity Rules

[SSM1]	Root Schema expressions MAY be split into multiple schema modules.
[SSM2]	A root schema in one namespace that is dependent upon type definitions or element declarations defined in another namespace MUST only import the root schema from that namespace.
[SSM3]	A root schema in one namespace that is dependant upon type definitions or element declarations defined in another namespace MUST NOT import internal schema modules from that namespace.
[SSM4]	All imported schema modules MUST be fully conformant with the Federal XML naming and design rules.
[SSM5]	Schema modules MUST either be treated as external schema modules or as internal schema modules of the root schema.
[SSM6]	All internal schema modules MUST be in the same namespace as their corresponding root schema.
[SSM7]	Each internal schema module MUST be named `ParentSchemaModule-Name}{InternalSchemaModuleFunction}{schema module}`
[SSM8]	A schema module defining all Federal `Common Complex Data Elements` MUST be created.
[SSM9]	The Federal Common Complex Data Elements schema module MUST be named "`fed:Common Complex Data Elements Schema Module`"
[SSM10]	A schema module defining all Federal `Common Simple Data Elements` MUST be created.

A.17 Schema Structure Modularity Rules

[SSM11]	The Federal `Common Simple Data Elements` schema module MUST be named "`fed:CommonSimpleDataElements Schema Module`"
[SSM15]	The Federal `Qualified Datatypes` schema module MUST be named "`fed:Qualified Datatypes schema module`"
[SSM16]	Agencies MAY create Agency level schema modules for reusable components not included in Federal level schema. Agencies SHOULD submit all Agency reusable components for consideration as Federal level reusable components.
[SSM17]	A schema module defining Agency MAY be created.
[SSM18]	Agency schema modules MUST be named ""
[SSM19]	A schema module defining all Agency MAY be created.
[SSM20]	Agency schema modules MUST be named ""
[SSM21]	A schema module defining all Agency Unqualified Datatypes MAY be created.
[SSM22]	Agency Unqualified Datatype schema modules MUST be named ""
[SSM23]	A schema module defining all Agency Qualified Datatypes MAY be created.
[SSM24]	Agency Qualified Datatypes schema modules MUST be named "

A.18 Standards Adherence rules

[STA1]	All schema design rules MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

[STA2]	All schema and messages MUST be based on the W3C suite of technical specifications holding recommendation status.
[STA3]	Proprietary extensions to the W3C specifications MUST never be used.

A.19 SimpleType Naming Rules

[STN1]	Each `xsd:simpleType` definition name MUST be the `datatype` dictionary entry name with the separators removed.

A.20 SimpleType Definition Rules

[STD1]	For every `datatype` whose metadata components map directly onto the properties of a built-in `xsd:DataType`, the `datatype` MUST be defined as a named `xsd:simpleType` in the `fed:unqualifiedDatatype` schema module.

A.21 Standards Rquirements Rule

[STR1]	To ensure conformance with both statutory and policy requirements contained in Public Law 104-113 and Office of Management and Budget Circular A-119, all Federal XML implementations must adhere to the following hierarchy of standards in creating and using XML ◆ De jure Voluntary Consensus Standards ◆ Cross-sector Voluntary Consensus Standards ◆ Sector specific Voluntary Consensus Standards ◆ Federal Enterprise Wide Standards

	◆ Agency specific standards
[STR2]	Agencies SHOULD create Agency level policy, procedures and guidance to ensure XML is developed and governed at an enterprise level
[STR3]	Agencies SHOULD promote Agency level XML components to candidate federal level components and candidate Voluntary Consensus Standards Bodies

A.22 Versioning Rules

[VER1]	Every federal and Agency Schema and schema module major version committee draft MUST have a document-id of the form `<name>-<major>.0[.<revision>]`
[VER2]	Every federal and Agency Schema and schema module major version Standard MUST have a document-id of the form `<name>-<major>.0`
[VER3]	Every minor version schema or schema module draft MUST have a document-id of the form `<name>-<major >.<non-zero>[.<revision>]`
[VER4]	Every minor version schema or schema module Standard MUST have an document-id of the form `<name>-<major >.<non-zero>`
[VER5]	For minor version changes, the name of the version construct MUST NOT change.

A.22 Versioning Rules

[VER6]	Every schema and schema module major version number MUST be a sequentially assigned, incremental number greater than zero.
[VER7]	Every schema and schema module minor version number MUST be a sequentially assigned, incremental non-negative integer.
[VER8]	A minor version document schema MUST import its immediately preceding version document schema.
[VER9]	Schema and schema module minor version changes MUST be limited to the use of xsd:extension or xsd:restriction to optionally alter existing types or add new constructs.
[VER10]	Schema and schema module minor version changes MUST not break semantic compatibility with prior versions.

Appendix B. Approved Acronyms and Abbreviations

The following Acronyms and Abbreviations have been approved for Federal and Agency use:

- A Dun & Bradstreet Data Universal Numbering System (DUNS) number *must* appear as "DUNS".
- "Identifier" *must* appear as "ID".
- "Uniform Resource Identifier" *must* appear as "URI"
- [Example] the "Uniform Resource. Identifier" portion of the **Binary Object. Uniform Resource. Identifier** attribute becomes "URI" in the resulting XML name). The use of URI for Uniform Resource Identifier takes precedence over the use of "ID" for "Identifier".

This list will henceforth be maintained by XXX, and additions included in current and future versions will be maintained and published separately.

Appendix C. Metadata Components

Table B-1. Approved Core Component Type Content and Supplementary Components

Name	Primitive data-type	Definition	Remarks
Amount Currency. Code List. Identifier	String	The Currency Code List being used.	Reference UN/ECE Rec. 9, using 3-letter alphabetic codes. The UN/ECE Rec. 9 is also published as ISO 4217, but is available in electronic form and free of charge.
Amount Currency. Code List Version. Identifier	string	The *Version* of the code list.	
Amount Currency. Identifier	string	The currency of the amount	The code identifying the currency
Binary Object. Format. Text	string	The format of the binary content.	
Binary Object. Mime.Code	string	The mime type of the binary object.	Reference IETF RFC 2045, 2046, 2047
Binary Object. Character Set. Code	string	The character set of the binary object if the mime type is text.	Reference IETF RFC 2045, 2046, 2047
Binary Object. Encoding. Code	string	Specifies the decoding algorithm of the binary object.	Reference IETF RFC 2045, 2046, 2047

Name	Primitive data-type	Definition	Remarks
Binary Object. Uniform Resource. Identifier	string	The Uniform Resource Identifier that identifies where the Binary Object is located.	
Binary Object. Filename. Text	String	The filename of the binary object.	Reference IETF RFC 2045, 2046, 2047
Code List. Agency. Identifier	string	An agency that maintains one or more code lists.	Defaults to the UN/EDIFACT data element 3055 code list.
Code List. Agency Name. Text	string	The name of the agency that maintains the code list.	
Code List. Name. Text	string	The name of a list of codes.	
Code List. Identifier	string	The identification of a list of codes	Can be used to identify the URL of a source that defines the set of currently approved permitted values
Code List Scheme. Uniform Resource. Identifier	string	The Uniform Resource Identifier that identifies where the code list scheme is located.	
Code List. Uniform Resource. Identifier	string	The Uniform Resource Identifier that identifies where the code list is located.	
Code List. Version. Identifier	string	The *Version* of the code list.	Identifies the *Version* of the UN/EDIFACT data element 3055 code list.

Name	Primitive data-type	Definition	Remarks
Code. Name. Text	string	The textual equivalent of the code content	If no code content exists, the code name can be used on its own
Date Time. Format. Text	string	The format of the date/time content	Reference ISO 8601 and W3C note on date time
Identification Scheme Agency. Identifier	string	The identification of the agency that maintains the identification scheme.	Defaults to the UN/EDIFACT data element 3055 code list.
Identification Scheme Agency. Name. Text	string	The name of the agency that maintains the identification scheme	
Identification Scheme Data. Uniform Resource. Identifier	string	The Uniform Resource Identifier that identifies where the identification scheme data is located	
Identification Scheme. Identifier	string	The identification of the identification scheme.	
Identification Scheme. Name. Text	string	The name of the identification scheme.	
Identification Scheme. Uniform Resource. Identifier	string	The Uniform Resource Identifier that identifies where the identification scheme is located.	
Identification Scheme. Version. Identifier	string	The *Version* of the identification scheme.	Identifies the *Version* of the UN/EDIFACT data element 3055 code list.

Name	Primitive data-type	Definition	Remarks
Indicator. Format. Text	String	Whether the indicator is numeric, textual or binary	
Language. Identifier	string	The identifier of the language used in the corresponding text string	Reference ISO 639: 1998
Language. Locale. Identifier	string	The identification of the locale of the language.	
Measure Unit. Code	string	The type of unit of measure	Reference UN/ECE Rec. 20 and X12 355.
Measure Unit. Code List Version. Identifier	string	The *Version* of the measure unit code list.	
Numeric. Format. Text	string	Whether the number is an integer, decimal, real number or percentage	
Quantity. Unit. Code	string	The unit of the quantity	May use UN/ECE Rec. 20
Quantity Unit. Code List Agency. Identifier	string	The identification of the agency which maintains the quantity unit code list	
Quantity Unit. Code List. Identifier	string	The quantity unit code list.	Defaults to the UN/EDIFACT data element 3055 code list.

Name	Primi-tive data-type	Definition	Remarks
Quantity Unit. Code List Agency Name. Text	string	The name of the agency which maintains the quantity unit code list.	

Appendix D. Permissible Representation Terms

Table C-1. Permissible Representation Terms

Primary Representation Term	**Definition**	**Related Unqualified Datatype**	**Secondary Representation Terms**
Amount	A number of monetary units specified in a currency where the unit of currency is explicit or implied.	Amount. Type	
Binary Object	A set of finite-length sequences of binary octets. [Note: This *Representation Term* shall also be used for *Data Types* representing graphics (i.e. diagram, graph, mathematical curves, or similar representation), pictures (i.e. visual representation of a person, object or scene), sound, video, etc.]	Binary Object. Type	Graphic, Picture, Sound, Video
Code	A character string (letters, figures or symbols) that for brevity and / or language independence may be used to represent or replace a definitive value or text of a *Property*. [Note: The term 'Code' should not be used if the character string identifies an instance of an *Object Class* or an object in the real world, in which case the *Representation Term* identifier should be used.]	Code. Type	
Date Time	A particular point in the progression of time (ISO 8601). [Note: This *Representation Term* shall also be used for *Data Types* only representing a Date or a Time.]	Date Time. Type	Date, Time

Primary Representation Term	**Definition**	**Related Unqualified Datatype**	**Secondary Representation Terms**
Identifier	A character string used to establish the identity of, and distinguish uniquely, one instance of an object within an identification scheme from all other objects within the same scheme.	Identifier. Type	
Indicator	A list of exactly two mutually exclusive Boolean values that express the only possible states of a *Property*. [Note: Values typically indicate a condition such as **`on/off`**; **`true/false`** etc.]	Indicator. Type	
Measure	A numeric value determined by measuring an object. Measures are specified with a unit of measure. The applicable unit of measure is taken from UN/ECE Rec. 20. [Note: This *Representation Term* shall also be used for measured coefficients (e.g. m/s).]	*Measure. Type*	
Numeric	Numeric information that is assigned or is determined by calculation, counting or sequencing. It does not require a unit of quantity or a unit of measure. [Note: This *Representation Term* shall also be used for *Data Types* representing Ratios (i.e. rates where the two units are not included or where they are the same), Percentages, etc.)	Numeric. Type	Value, Rate, Percent

Primary Representation Term	**Definition**	**Related Unqualified Datatype**	**Secondary Representation Terms**
Quantity	A counted number of non-monetary units. Quantities need to be specified with a unit of quantity. [Note: This *Representation Term* shall also be used for counted coefficients (e.g. flowers/m²).]	Quantity. Type	
Text	A character string (i.e. a finite set of characters) generally in the form of words of a language. [Note: This *Representation Term* shall also be used for names (i.e. word or phrase that constitutes the distinctive designation of a person, place, thing or concept).]	Text. Type	Name

Secondary Representation Terms will also be defined as unqualified datatypes.

Appendix E. Technical Terminology

Ad hoc schema processing	Doing partial schema processing, but not with official schema validator software; e.g., reading through schema to get the default values out of it.
Aggregated Data Elements	A collection of related pieces of data. Expressed in modeling terms, it is the representation of an Object Class, in a specific Business Context.
Application-level validation	Adherence to business requirements, such as valid account numbers.
Assembly	Using parts of the library of reusable components to create a new kind of business document type.
Business Context	Defines a context in which a business has chosen to employ an information entity.
Business Object	An unambiguously identified, specified, referenceable, registerable and re-useable scenario or scenario component of a business transaction. The term business object is used in two distinct but related ways, with slightly different meanings for each usage: In a business model, business objects describe a business itself, and its business context. The business objects capture business concepts and express an abstract view of the business's "real world". The term "modeling business object" is used to designate this usage. In a design for a software system or in program code, business objects reflects how business concepts are represented in software. The abstraction here reflects the transformation of business ideas into a software realization. The term "systems business objects" is used to designate this usage.

business semantic(s)	A precise meaning of words from a business perspective.
Business Term	This is a synonym under which a data element is commonly known and used in the business. A data element or data element aggregation may have several business terms or synonyms.
class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment. See interface.
class diagram	Shows static structure of concepts, types, and classes. Concepts show how users think about the world; types show interfaces of software components; classes show implementation of software components. (OMG Distilled) A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships. (Rational Unified Process)
Common attribute	An attribute that has identical meaning on the multiple elements on which it appears. A common attribute might or might not correspond to an XSD global attribute.
component	One of the individual entities contributing to a whole.
context	Defines the circumstances in which a Business Process may be used.
context category	A group of one or more related values used to express a characteristic of a business circumstance.
data centric	

data element	
document centric	
Root schema	A schema document corresponding to a single name-space, which is likely to pull in (by including or import-ing) schema modules.
Datatype	A descriptor of a set of values that lack identity and whose operations do not have side effects. Datatypes include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations. (XSD) (ISO 11179)
Generic BIE	A semantic model that has a "zeroed" context. We are assuming that it covers the requirements of 80% of business uses, and therefore is useful in that state.
instance	An individual entity satisfying the description of a class or type.
Instance constraint checking	Additional validation checking of an instance, beyond what XSD makes available, that relies only on con-straints describable in terms of the instance and not ad-ditional business knowledge; e.g., checking co-occurrence constraints across elements and attributes. Such constraints might be able to be described in terms of Schematron.
Intermediate element	An element not at the top level that is of a complex type, only containing other elements and attributes.

Internal schema module:	A schema module that does not declare a target name-space.
Leaf element	An element containing only character data (though it may also have attributes). Note that, because of the XSD mechanisms involved, a leaf element that has attributes must be declared as having a complex type, but a leaf element with no attributes may be declared with either a simple type or a complex type.
Lower-level element	An element that appears inside a business message. Lower-level elements consist of intermediate and leaf level.
Object Class	The logical data grouping (in a logical data model) to which a data element belongs (ISO11179). The *Object Class* is the part of a *simple or complex data elements Dictionary Entry Name* that represents an activity or object in a specific *Context*.
Namespace schema module:	A schema module that declares a target namespace and is likely to pull in (by including or importing) schema modules.
Naming Convention	The set of rules that together comprise how the dictionary entry name for simple data elements; complex data elements; and XSD elements, attributes and types are constructed.
(XML) Schema	An XML Schema consists of components such as type definitions and element declarations. These can be used to assess the validity of well-formed element and attribute information items (as defined in [XML-Infoset]), and furthermore may specify augmentations to those items and their descendants.
Schema module	A collection of XML constructs that together constitute an XSD conformant schema. Schema modules are intended to be used in combination with other XSD conformant schema.

Schema Processing	Schema validation checking plus provision of default values and provision of new infoset properties.
Schema Validation	Adherence to an XSD schema.
semantic	Relating to meaning in language; relating to the connotations of words.
Top-level element	An element that encloses a whole business message. Note that business messages might be carried by messaging transport protocols that themselves have higher-level XML structure. Thus, a top-level element is not necessarily the root element of the XML document that carries it.
type	Description of a set of entities that share common characteristics, relations, attributes, and semantics. A stereotype of class that is used to specify an area of instances (objects) together with the operations applicable to the objects. A type may not contain any methods. See class, instance. Contrast interface.