SISO-REF-056-2018

SISO Reference for XML Style Guide

1 August 2018

SAC Approved: 07/11/2018

EXCOM Approved: 08/01/2018

Prepared by:
Simulation Interoperability Standards Organization
Standards Activity Committee (SAC)
Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Section</th>
<th>Date (MM/DD/YYYY)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SISO-REF-056-2018</td>
<td>All</td>
<td>08/01/2018</td>
<td>First version</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

1 INTRODUCTION .......................................................................................................................... 7
   1.1 PURPOSE ............................................................................................................................... 7
   1.2 SCOPE ................................................................................................................................. 7
   1.3 OBJECTIVES ....................................................................................................................... 8
   1.4 INTENDED AUDIENCE ...................................................................................................... 8

2 REFERENCES ............................................................................................................................. 8
   2.1 SISO REFERENCES ............................................................................................................. 8
   2.2 OTHER REFERENCES ........................................................................................................ 8

3 DEFINITIONS ............................................................................................................................. 9

4 ACRONYMS AND ABBREVIATIONS ......................................................................................... 9

5 DOCUMENT CONVENTIONS ..................................................................................................... 10
   5.1 FORMATTING ..................................................................................................................... 11
   5.2 TERMINOLOGY ................................................................................................................... 11
      5.2.1 XML Information Set Terminology ............................................................................ 11
      5.2.2 XML Schema Terminology ...................................................................................... 12
      5.2.3 XML Namespace Terminology .................................................................................. 12
   5.3 DOCUMENT ORGANIZATION ............................................................................................ 12

6 GUIDING PRINCIPLES ............................................................................................................. 12
   6.1 XML SCHEMA DESIGN PRINCIPLES .............................................................................. 13
      6.1.1 Disallow Content Modification with XML Processors ............................................... 13
      6.1.2 Use XML Validating Parsers for Content Validation ............................................... 13
      6.1.3 Validate for Conformance ....................................................................................... 13
      6.1.4 Allow Multiple XML Schemas for XML Constraints ............................................ 14
      6.1.5 Disallow Mixed Content ......................................................................................... 14
      6.1.6 Specify Types for All Constructs ............................................................................ 14
      6.1.7 Avoid Wildcards ....................................................................................................... 14
      6.1.8 Use Open Standards ............................................................................................... 14
   6.2 NAMESPACE PRINCIPLES .................................................................................................. 14
   6.3 MODELING PRINCIPLES .................................................................................................... 15
      6.3.1 Consistent Naming ...................................................................................................... 15
      6.3.2 Reflect the Real World ............................................................................................ 16
      6.3.3 Be Consistent ........................................................................................................... 16
      6.3.4 Reserve Inheritance for Specialization ..................................................................... 16
      6.3.5 Do Not Duplicate Definitions ................................................................................... 16
      6.3.6 Keep It Simple .......................................................................................................... 16
      6.3.7 Be Mindful of Namespace Cohesion ......................................................................... 17

7 RELATION TO STANDARDS .................................................................................................... 17
   7.1 XML 1.0 ............................................................................................................................. 17
   7.2 XML NAMESPACES .......................................................................................................... 17
   7.3 XML SCHEMA ................................................................................................................. 17
   7.4 DOCUMENTED COMPONENTS ........................................................................................ 18

8 XML SCHEMA DESIGN BEST PRACTICES .............................................................................. 19
   8.1 RESTRICTIONS ON XML SCHEMA CONSTRUCTS ....................................................... 19
      8.1.1 No Mixed Content....................................................................................................... 19
      8.1.2 No Notations ............................................................................................................. 19
8.1.3 XML Schema Inclusion .......................................................... 19
8.1.4 No XML Schema Redefinition ............................................ 20
8.1.5 Wildcard Restrictions .......................................................... 20
  8.1.5.1 No Unconstrained Type Substitution ................................ 20
  8.1.5.2 No Unconstrained Text Substitution ................................ 20
  8.1.5.3 Untyped Elements Should Be Abstract .............................. 20
  8.1.5.4 No Untyped Attributes .................................................. 21
  8.1.5.5 No Unconstrained Attribute Substitution ......................... 21
8.1.6 Component Naming Restrictions ......................................... 21
  8.1.6.1 No Anonymous Type Definitions .................................. 21
  8.1.6.2 No Local Element Declarations .................................... 21
  8.1.6.3 No Local Attribute Definitions .................................... 22
8.1.7 No Uniqueness Constraints ................................................. 22
8.1.8 Model Group Restrictions .................................................. 22
  8.1.8.1 No Recursively Defined Model Groups ............................ 22
  8.1.8.2 XML Schema Particle Cardinality Restrictions ................ 23
  8.1.8.3 Default Value Restrictions .......................................... 23
8.1.9 Block Substitution Restrictions .......................................... 23
8.1.10 Final Value Restrictions .................................................. 24
8.2 XMLSCHEMA XSD:SCHEMA DOCUMENT ELEMENT ... 24
8.3 NAMESPACE IMPORTS ............................................................. 25
  8.3.1 xsd:import Element Restrictions .................................... 25
  8.3.2 Including XML Content from Other Namespaces .................... 26
8.4 ANNOTATIONS ................................................................. 27
  8.4.1 Human-Readable (Natural Language) Documentation ............... 27
  8.4.2 Machine-Readable Annotations ....................................... 28
8.5 COMMENTS ................................................................. 29
8.6 TYPE DEFINITIONS .......................................................... 29
  8.6.1 Complex Type Definitions ............................................. 29
  8.6.2 Simple Content (CSC) Restrictions .................................. 29
  8.6.3 XML Schema Complex Content (CCC) Restrictions ............... 30
8.7 ADDITIONAL DEFINITIONS AND DECLARATIONS ....................... 30
  8.7.1 Element Declarations .................................................. 30
  8.7.2 Attribute Declarations ................................................ 30
9 XML SCHEMA MODELING BEST PRACTICES .................................. 31
  9.1 XSD:SCHEMA DOCUMENT ELEMENT RESTRICTIONS ................. 31
  9.2 ANNOTATIONS ............................................................. 31
    9.2.1 Human-Readable (Natural Language) Documentation ............. 31
  9.3 SIMPLE TYPE DEFINITIONS .............................................. 32
  9.4 COMPONENT USAGE ...................................................... 32
    9.4.1 Reference Elements ................................................ 33
  9.5 XML SCHEMA USING EXTERNAL XML SCHEMAS ......................... 34
  9.6 XML SCHEMA CONTAINER ELEMENTS .................................... 36
10 XML SCHEMA NAMING BEST PRACTICES .................................... 37
  10.1 EXTENSION OF XSD NAMESPACE SIMPLE TYPES ......................... 37
  10.2 USAGE OF ENGLISH ...................................................... 37
  10.3 CHARACTERS IN NAMES ................................................ 37
  10.4 CHARACTER CASE ...................................................... 38
  10.5 CHARACTER SETS ..................................................... 38
  10.6 USE OF ACRONYMS AND ABBREVIATIONS ................................ 38
11 INSTANCE (SAMPLE) XML DOCUMENTS ...................................... 39
List of Figures

FIGURE 1: EXAMPLE OF AN XML FRAGMENT ............................................................................................................. 11
FIGURE 2: EXAMPLE OF THE USE OF A NAMESPACE ................................................................................................. 15
FIGURE 3: USE OF EXTERNAL COMPONENTS TO CREATE AN XML TYPE .......................................................... 35
FIGURE 4: EXAMPLE OF CONTAINER ELEMENT ......................................................................................................... 36

List of Tables

TABLE 3. TERMS AND DEFINITIONS ............................................................................................................................... 9
TABLE 4. ACRONYMS AND ABBREVIATIONS .................................................................................................................. 9
1 INTRODUCTION
The Extensible Markup Language (XML) has become a ubiquitous format for the structure and content of data in software applications and across the World Wide Web. Several standards products developed by the Simulation Interoperability Standards Organization (SISO) use XML to specify the structure and content of data for storing and exchanging information (e.g., Base Object Model, Military Scenario Definition Language, Coalition Battle Management Language, Federation Engineering Agreements Template). These products were developed independently by SISO groups having diverse experience with XML, resulting in different approaches to the use of the language features. In parallel, major organizations, such as the U.S. Government’s National Information Exchange Model (NIEM) program, have developed guidelines, called Naming and Design Best Practices (NDBP), for the use of XML. Such guidelines are intended to facilitate wide-spread understanding and employment while applying a common style [3]. To apply a similar approach to create commonality across XML-based SISO products, this XML Schema NDBP provides guidance and best practices for naming and designing components of XML schemas used in SISO products.

This SISO XML Style Guide (XSG) specifies a set of best practices for defining standard information exchange messages, transactions, and documents on a large scale across multiple communities of interest. These reusable components are rendered in XML schema documents as type, element, and attribute definitions that comply with the W3C XML Schema [4] specification. The resulting XML schemas are available to Modeling and Simulation (M&S) practitioners and developers at: http://www.sisostds.org/schemas/

The W3C XML Schema [4] standard enables information interoperability and sharing by providing a common language for describing data precisely. The constructs it defines are basic metadata building blocks — baseline data types and structural components. Users employ these building blocks to describe their own domain-specific data semantics and structures, as well as structures for specific information exchanges and components for reuse across multiple information exchanges. Best practices that profile allowable XML Schema constructs and describe how to use them help ensure that those components are consistent and reusable.

1.1 Purpose
The purpose of this document is to inform SISO product developers about best practices for the style of XML documents when XML is used to specify the structure and content of data supporting a SISO standard. This document is a reference that defines categories of principles and best practices to follow. Adoption of these principles and practices is voluntary, but groups operating within SISO are expected to expend reasonable effort to understand the principles and to apply them to their products unless doing so would substantively reduce the effectiveness of those products in supporting SISO objectives.

1.2 Scope
This document was developed to specify XML schema best practices as they apply to SISO Standards, Guidance, and Reference Products. The document covers the following issues in depth:

- Guiding principles behind the design of XML schemas
- Best practices for using XML schema constructs in SISO products
- Best practices for modeling and structuring XML schemas
- Best practices for creating XML instances
- Best practices for naming XML components
- Best practices for extending XML components

The best practices, principles and definitions contained in this document are intended to communicate the technical details of the XML Schema language for use in specifying information exchange formats in XML.
1.3 Objectives

SISO develops, manages, maintains, and promulgates user-driven Modeling and Simulation (M&S) standards that improve the technical quality and cost efficiency of M&S implementations across the worldwide M&S community. The objectives of technical quality and cost efficiency are to:

- Promote consistency in how XML Schema products are designed.
- Facilitate XML product development and reduce authoring effort for the SISO product.
- Enable automated publication of SISO XML products in natural human-readable formats that do not require XML expertise to understand.
- Provide a decision tree for SISO groups to deliberately use to identify and tailor the best practices they intend to follow in developing their XML products.

1.4 Intended Audience

This document targets diverse M&S communities, including developers, procurers, and users worldwide who employ SISO XML products for information exchange and interoperability. Such information exchanges may be between or within organizations.

This document specifies principles and best practices for SISO XML data components and XML schemas. The scope of this document does not include mandatory requirements to be followed by SISO groups in the development of their XML products.

2 REFERENCES

2.1 SISO References

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SISO-ADM-001-2014</td>
<td>Policy for Numbering of SISO Products</td>
</tr>
<tr>
<td>2 SISO-ADM-005-2011</td>
<td>Policy for the Style and Format of SISO Documents</td>
</tr>
</tbody>
</table>

2.2 Other References

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 W3C XML Schema</td>
<td>XML Schema <a href="http://www.w3.org/XML/Schema">Link</a></td>
</tr>
<tr>
<td>7 NOTE-swbp-n-aryRelations-20060412</td>
<td>Defining N-ary Relations on the Semantic Web, W3C Working Group Note 12 April 2006, <a href="http://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/">Link</a>. Use Case 3 is described at Use Case 3: N-ary relation with no distinguished participant</td>
</tr>
</tbody>
</table>
### Table 1. Terms and Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adapter type</strong></td>
<td>An adapter type is an XML type that adapts external components for use within SISO XML Guidance. An adapter type creates a new class of object that embodies a single concept composed of external components. Adapter types avoid namespace collisions when including multiple schemas.</td>
</tr>
<tr>
<td><strong>Data definition</strong></td>
<td>The data definition of a <strong>documented component</strong> is the content of the first occurrence of the XML Schema element <code>xsd:documentation</code>, which is an immediate child of an occurrence of the XML Schema element <code>xsd:annotation</code>, which is an immediate child of the element that defines the component (where “xsd:” is a namespace prefix referring to the W3C XML Schema namespace).</td>
</tr>
<tr>
<td><strong>Documented component</strong></td>
<td>A documented component is an XML schema component that has an associated <strong>data definition</strong>. These XML schema components have a textual definition, so that the component may be well-understood.</td>
</tr>
</tbody>
</table>

### Table 2. Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Merriam-Webster's Collegiate Dictionary, 11th Edition</td>
</tr>
</tbody>
</table>
5 DOCUMENT CONVENTIONS

This document uses formatting and syntactic conventions to clarify meaning and avoid ambiguity. These conventions build upon those specified in SISO-ADM-005.

This document relies on references to many outside documents. Such references are noted by bracketed inline terms. For example, a reference to ISO/IEC 11179-4:2004 is shown as [5].

Principles and best practices are documented as follows:

[Principle <number>]

The principles represent the concepts and goals that have helped shape the SISO XSG. Principles act as the basis on which the best practices are defined.

Accompanying each principle is a short discussion section that justifies the application of the principle to SISO schema design.

Principles are numbered in the order in which they appear in the document.

[Best Practice <section>-<number>]

This identifies a best practice for schemas (e.g., using techniques such as Schematron, “a language for making assertions about patterns found in XML documents”; see http://www.schematron.com/).

Best practices state specific best practices for artifacts, such as XML schemas and instances.

Best practices are stated using both XML Information Set (InfoSet) terminology (elements and attributes) [12] and XML Schema terminology (schema components) [4]. The choice of terminology is driven by which standard best expresses the best practices. Certain concepts are more clearly expressed using XML InfoSet information items, others using the XML Schema data model; still others are best expressed using a combination of terminology drawn from both standards.

Best practices have rationales that justify the need for the best practice. For clarity, there may be multiple best practices that have the same rationale.

Best practices and supporting text may use Extended Backus-Naur Form (EBNF) notation as defined by [11].
Best practices are numbered according to the section in which they appear and the order in which they appear within that section. For example, [Best practice 9-1] is the first best practice in Section 9-
XML Schema Modeling Best Practices.

Each best practice is accompanied by a description of its applicability. This identifies the type of XML
schema to which the best practice applies or indicates whether the best practice is applicable to XML
documents or element information items.

5.1 Formatting

In addition to special formatting for definitions, principles, and best practices, this document uses
consistent formatting to identify SISO XML components.

Courier: All words appearing in Courier font are values, objects, keywords, or literal XML text.

Italics: All words appearing in italics, when not titles or used for emphasis, are special terms with
definitions appearing in this document.

Keywords: Keywords reflect concepts or constructs expressed in the language of their source standard.
Keywords have been given an identifying prefix to reflect their source. The following prefixes are used:


Throughout the document, fragments of XML schemas or XML instances are used to clarify a principle or
best practice. These fragments are specially formatted in Courier font and appear in text boxes. An
example of such a fragment follows:

```xml
<xsd:complexType name="ObjectType">
  ... 
</xsd:complexType>
```

Figure 1: Example of an XML Fragment

5.2 Terminology

This document uses standard terminology to explain the principles and best practices that describe XML
Schemas.

5.2.1 XML Information Set Terminology

This document uses the concepts of element information items (“element”), attribute information items
(“attribute”), and their associated properties as defined by [12] with clarifications as discussed below.
Note that in the clarification that follows, the abstract property names appear in square brackets adjacent
to the information items to which they belong. For example, “Element[parent]” discusses the abstract
property “parent” of the element information item.

- parent of an element (Element[parent])
- child of an element (Element[children])

Note that the InfoSet properties “Element[parent]” and “Element[children]” correspond to a direct,
immediate relationship with an element. Children of an element and their children, and so on, are
collectively referred to as descendants of that element. Parents of an element and their parents,
and so on, are collectively referred to as ancestors of that element.

- element owning an attribute (Attribute[owner element])
  The owner of an attribute is the element that possesses or contains the attribute.
5.2.2 XML Schema Terminology

The terms W3C XML Schema [4], XML Schema (upper case XML Schema), and XSD all refer to the XML Schema definition language, as specified in the two-part XML Schema specification:

- XML Schema Part 2: Datatypes [16]

The term XML schema (lower case schema) refers to specific XML schema documents that conform to the XML Schema specifications listed above.

The terms XML instance and XML document refer to an XML instance document, which is defined by and validates to a particular XML schema (when one is referenced by the instance document).

The term schema component is defined in [4] as a building block for XML Schema. This document refers to, rather than restates, the definitions of the different XML schema components associated with the XML Schema Abstract Data Model, which are defined in the XML Schema specification. In this document, the name of the referenced XML schema component may appear without the suffix “schema component” (e.g., the term “complex type definition” may be used instead of “complex type definition XML schema component”) to enhance readability of the text.

The term NCName is defined in [16] and refers to XML noncolonized names, which are XML name strings that do not contain the “:” character; in contrast to qualified names (QName) such as xsd:documentation.

5.2.3 XML Namespace Terminology

This document uses the concept of an XML Namespace as defined by [13] and [14].

5.3 Document Organization

This remainder of this document is organized into sections as follows:

- Section 6 - Guiding Principles - discusses the principles that serve as the foundation of and guidance for the best practices.
- Section 7 - Relation to Standards - discusses the use of the key standards used in the development of SISO this guide.
- Section 8 XML Schema Design - presents best practices for using XML Schema constructs in XML schemas.
- Section 9 - XML schema Modeling - discusses the best practices for additional structures and constraints needed to build XML schemas.
- Section 10 XML Schema Naming - discusses the best practices used in naming data components.
- Section 11 - Instance (Sample) XML Documents - discusses providing sample XML documents to accompany XML schemas.
- Section 12 - XML Schema Naming - provides information on locating SISO XML documents.

NOTE: The ordering of the sections is intended to minimize the number of forward references in the document. For this reason, the naming best practices appear in one of the last sections of the document, so that the concepts being named have already been discussed.

6 GUIDING PRINCIPLES

Principles in this section provide a foundation for the best practices. These principles are generally applicable in most cases. They should not be used as a replacement for common sense or appropriate special cases.

The principles discussed in this section are categorized as follows:
6.1 XML Schema Design Principles

The principles in this section address how XML Schema technology should be used in designing XML schemas and instances.

6.1.1 Disallow Content Modification with XML Processors

XML Schema has constructs that can make the data provided by XML processors different before and after XML schema processing. An example of this is the use of XML Schema attribute declarations with default values. Before XML schema validation, there may be no attribute value, but after processing, the attribute value exists.

The purpose of processing instances against XML schemas is solely validation: testing that XML document instances match desired structure and content specified in the associated XML schema(s). It should not be used to change the content of data instances.

[Principle 1] The content of a data instance SHOULD NOT be modified through the process of validation against the associated XML schema document(s).

6.1.2 Use XML Validating Parsers for Content Validation

XML schemas should be designed for XML Schema validation. A primary goal is to maximize the amount of validation that may be performed by XML Schema validating parsers.

XML schemas validate content using content models: descriptions of what elements and attributes may be contained within an element, and what values are allowable. It is the XML element hierarchy (elements with attributes and unstructured content, contained by other elements) that XSD specifies and that XML Schema validating parsers can validate.

Mechanisms involving linking using attribute and element values are useful, but they should only be relied on when absolutely necessary, as XML Schema validating parsers cannot readily validate them. For example, if a link is established via attribute values, an XML Schema validating parser cannot determine that participants have appropriate type definitions. Whenever possible, XML content should rely on XML syntax that can be validated with accompanying XML schema document(s).


6.1.3 Validate for Conformance

Systems that operate on XML data have the opportunity to perform multiple layers of processing. Middleware, XML libraries, XML schemas, and application software may process data. The primary purpose of XML Schema validation is to restrict processed data to that which conforms to agreed-upon best practices. This restriction is achieved by marking as invalid data that does not conform to the best practices defined by associated XML schema.

[Principle 3] Systems that use conformant data SHOULD mark as invalid data that does not conform to the best practices defined by applicable XML schema documents.

6.1.4 Allow Multiple XML Schemas for XML Constraints

SISO XML schemas should not be designed as one-size-fits-all to perform all validation. Instead, XML development groups should create a set of XML schemas, on which additional constraints may be placed.
XML development groups should not focus on language-binding XML schema implementations, which convert XML schema definitions into working programs, but rather focus on normalizing language and preserving the meaning of data.

[Principle 4]
Constraints on XML instances MAY be validated by multiple XML schema validation passes, using multiple XML schemas for a single namespace.

6.1.5 Disallow Mixed Content

XML data that use mixed content are difficult to specify and complicate the task of data processing. Much of the payload carried by mixed content is unchecked and does not facilitate data standardization or validation.

[Principle 6]
SISO XML schemas SHOULD NOT specify data that use mixed content.

6.1.6 Specify Types for All Constructs

XML schema components should include defined names. This means that there are no anonymous types, elements, or other components. Once an application has determined the name (i.e., namespace and local name) of an attribute or element used in instances, it will also know the type of that attribute or element.

XML schema components should include only global attributes and elements (no local definitions). This maximizes the ability of application developers to extend, restrict, or otherwise derive definitions of components. Using named global components in XML schemas maximizes the capacity for reuse.

[Principle 7]
SISO XML schemas SHOULD NOT use or define local or anonymous components, as such definitions do not facilitate reuse of all schema components.

6.1.7 Avoid Wildcards

Wildcards work in opposition to standardization. The goal of creating harmonized, standard XML schemas is to standardize definitions of data. The use of wildcard mechanisms (such as xsd:any, which allows insertion of an arbitrary number of elements from any namespace) allows nonstandard data to be passed via otherwise standardized exchanges.

[Principle 8]
XML components SHOULD NOT incorporate wildcards unless absolutely necessary, as they hinder standardization by encouraging use of non-standardized data rather than standardized data.

6.1.8 Use Open Standards

The cooperative efforts of many knowledgeable individuals have resulted in published information standards. Where appropriate and applicable, XML development groups should leverage these standards.

[Principle 9]
XML standards and schemas SHOULD leverage and enable use of other open standards.

6.2 Namespace Principles

The principles in this section addresses the design philosophy used in designing a conceptual model of an XML schema.

XML schemas should be designed to maximize reuse of namespaces and other XML schemas that define them. When referring to an XML concept defined by SISO, a user should ensure that instances and XML schemas refer to the namespace defined by the SISO XML products. User-defined namespaces
should be used for specializations and extension of SISO XML product constructs, but should not be used when the SISO XML product structures are sufficient.

[Principle 10]

XML instances and XML schemas SHOULD reuse components from SISO distribution XML schemas when possible.

XML interoperability and reuse depend heavily on XML namespaces to prevent naming conflicts. Reuse of any component is always by reference to both its namespace and its local name. All XML component names should have global scope.

Example:

```
<xsd:element ref="nc:BinaryCaptureDate" minOccurs="0" maxOccurs="unbounded"/>
```

Figure 2: Example of the Use of a Namespace

In this example, nc:BinaryCaptureDate is reused by referencing its element declaration through both its namespace (which is bound to the prefix nc:) and its local name (BinaryCaptureDate). If an element named BinaryCaptureDate is declared in another namespace, it is an entirely different element than nc:BinaryCaptureDate. There is no implicit relationship to nc:BinaryCaptureDate.

From a business perspective, the two elements are likely to be related in the sense that they may have very similar semantic meanings. They may have essentially the same meaning, but slightly different properties. Such a relationship may commonly exist. However, any relationship between the two elements should be made explicit using methods outlined in this document.

[Principle 11]

A component SHOULD be identified by its local name together with its namespace. A namespace SHOULD be a part of the name of a component. A component's local name SHOULD NOT imply a relationship to components with similar names from other namespaces.

6.3 Modeling Principles

This XSG specifies components, XML schemas, and instances. The principles in this section address decisions that data modelers may face when creating SISO XML schema representations of domain data. It may not be possible to apply all guidelines in every case.

6.3.1 Consistent Naming

XML components should be given names that are consistent with names of other XML components. Having consistent names for components has several advantages:

1. It is easier to determine the nature of a component when it has a name that conveys the meaning and use of the component.
2. It is easier to find a component when it is named predictably.
3. It is easier to create a name for a component when clear guidelines exist.

[Principle 12]

Components SHOULD be given names that are consistent with names of other XML components. Such names SHOULD be based on simple best practices.

6.3.2 Reflect the Real World

SISO develops, manages, maintains, and promulgates user-driven M&S standards that improve the technical quality and cost efficiency of M&S implementations across the world-wide M&S community. To help facilitate unambiguous understanding of SISO reusable XML components, the names and structures...
should represent and model the informational aspects of objects and concepts that users are most familiar with. Types should not simply model collections of data.

[Principle 13]
Component definitions in SISO XML schemas SHOULD reflect real-world concepts.

6.3.3 Be Consistent
There should be no conflicts of meaning among types. This holds for types within a namespace, as well as types in different namespaces. A type should be used consistently in similar situations for similar purposes. Types should be defined for clear understanding and ease of intended use.

[Principle 14]
Component definitions in SISO XML schemas SHOULD have semantic consistency.

6.3.4 Reserve Inheritance for Specialization
Specialization should not be applied simply for the sake of achieving property inheritance. Specialization should be applied only where it is meaningful and appropriate to model permanent sibling subclasses of a base class that are mutually exclusive of one another.

[Principle 15]
Complex type definitions in SISO XML schemas SHOULD use type inheritance only for specialization.

Note that application of type augmentations is a well-defined exception to this guideline.

6.3.5 Do Not Duplicate Definitions
A real-world entity should be modeled in only one way. The definition of a type or element should appear once and only once. Multiple components of identical or closely similar semantics hinder interoperability because too many valid methods exist for representing the same data. For each data concept represented, there should be only one component (and associated type) to represent it.

Components with very similar semantics may exist in different contexts. For example, a complex type created for a particular exchange may appear to have identical or closely similar semantics to a complex type defined in another XML schema, or XML schema subset of the SISO product being authored. However, the type defined at the exchange level will have much more precise business requirements and syntax, compared with the broad definitions that are heavily reused. Specific contextual definitions should be considered semantic changes. This includes the application of augmentations to create a specialized type for a specific use.

Two components may have the same definition while having different representations. For example, a string may hold the complete name of a person, or the name may be represented by a structure that separates the components of the name into first, last, etc. The definition of alternative representations should not be considered duplication.

[Principle 16]
Multiple components with identical or undifferentiated semantics SHOULD NOT be defined. Component definitions SHOULD have clear, explicit distinctions.

6.3.6 Keep It Simple
All SISO XML content and structure is fundamentally based on business requirements for information exchange.

[Principle 17]
SISO XML schemas SHOULD have the simplest possible structure, content, and architecture consistent with real business requirements.
6.3.7 Be Mindful of Namespace Cohesion

Namespaces should maximize cohesion. The namespace methodology helps prevent name clashes among communities or domains that have different business perspectives and may choose identical data names to represent different data concepts. A namespace should be designed so that its components are consistent, may be used together, and may be updated at the same time.

[Principle 18]
XML namespaces defined by SISO XML schemas SHOULD encapsulate data components that are coherent, consistent, and internally related as a set. A namespace SHOULD encapsulate components that tend to change together.

7 RELATION TO STANDARDS

This section specifies the standards and specifications used in the development of this XSG. Where this XSG differs from public standards, the rationale for those differences is discussed in this section. The complete list of standards and specifications referenced in this section appears in Section 2 - References.

7.1 XML 1.0

[Best Practice 7-1]
SISO XML schemas SHOULD conform to XML as specified by [12].

Rationale
XML is a well-known, commonly used W3C Recommendation. It is supported by a large number of commercial and open-source software tools. It is a simple, well-defined, semi-structured data format that is flexible enough to allow for easy extension. XML works with many other powerful associated technologies such as XML Schema, XSLT, and XPath. XML artifacts of SISO should conform to the most recent W3C recommendation for XML.

7.2 XML Namespaces

[Best Practice 7-2]
SISO XML schemas SHOULD conform to the specification for namespaces in XML, as defined by [13] and [14].

Rationale
SISO is the organization dedicated to the promotion of modeling and simulation interoperability and reuse for the benefit of diverse M&S communities, including developers, procurers, and users worldwide. The ultimate scope of SISO XML standards is anticipated to be quite large. The primary purpose of namespaces is to avoid naming conflicts. Therefore, in SISO, XML namespaces are employed both to avoid name conflicts and to provide a level of independence to participating domains.

7.3 XML Schema

[Best Practice 7-3]

Rationale
XML Schema has become the generally accepted XML schema language and is experiencing the most widespread adoption. Although other XML schema languages exist that offer their own
advantages and disadvantages, the current approach is to base XML schemas on the W3C XML Schema [4] specification.

7.4 Documented Components

Good data definitions are fundamental to data interoperability. XML schemas should employ the guidance of [4] as a baseline for its data component definitions. All XML components are documented.

[Best Practice 7-4]

Within a SISO XML schema, the data definition provided for each documented component SHOULD follow the requirements and recommendations for data definitions given by [4].

Rationale

To advance the goal of creating semantically rich SISO XML schemas, it is necessary that data definitions be descriptive, meaningful, and precise. [4] provides standard structure and best practices for defining data definitions. SISO should use this structure for component definitions.

Note that the metadata maintained for each XML component contains additional details, including domain-specific usage examples and keywords. Such metadata are used to enhance search and discovery of components in a registry, and therefore, is not included in XML schemas.

For convenience and reference, the summary requirements and recommendations in [5] are reproduced here:

ISO 11179 Requirements

A data definition SHOULD:

- Be stated in the singular.
- State what the concept is, not only what it is not.
- Be stated as a descriptive phrase or sentence(s).
- Contain only commonly understood abbreviations.
- Be expressed without embedding definitions of other data or underlying concepts.

ISO 11179 Recommendations

A data definition SHOULD:

- State the essential meaning of the concept.
- Be precise and unambiguous.
- Be concise.
- Be able to stand alone.
- Be expressed without embedding rationale, functional usage, or procedural information.
- Avoid circular reasoning.
- Use the same terminology and consistent logical structure for related definitions.
- Be appropriate for the type of metadata item being defined.

In addition to the requirements and recommendations of [5], this XSG applies additional best practices to data definitions. These best practices are detailed in Section 9.2.1, Human-Readable (Natural Language) Documentation.

8 XML SCHEMA DESIGN BEST PRACTICES

The W3C XML Schema [4] Language provides many features that allow a developer to represent a logical data model many different ways. This section establishes best practices for the use of XML schema constructs within SISO XML schemas. Because the XML schema specifications are flexible,
comprehensive best practices are needed to achieve a balance between establishing uniform XML schema design and providing developers flexibility to solve novel data modeling problems.

The XML schema design best practices in this section fall into the following categories:

- Restrictions on XML Schema Constructs
- `xsd:schema` Document Element
- Namespace Imports
- Annotations
- Type Definitions
- Additional Definitions and Declarations

8.1 Restrictions on XML Schema Constructs

A number of XML schema constructs should not be used. Some of these constructs create problems for interoperability, with tool support, or with clarity or precision of data model definition.

8.1.1 No Mixed Content

[Best Practice 8-1]

Within a SISO XML schema, an element `xsd:complexType` SHOULD NOT own the attribute `mixed` with the value `true`.

[Best Practice 8-2]

Within a SISO XML schema, an element declaration that is of complex content SHOULD NOT own the attribute `mixed` with the value `true`.

Rationale

Mixed content allows the mixing of data tags with text. Languages such as XHTML use this syntax for markup of text. Mixed content creates complexity in processing, defining, and constraining content. Well-defined markup languages exist outside of XML and may be used with XML data. External XML schemas may include mixed content.

8.1.2 No Notations

[Best Practice 8-3]

A SISO XML schema SHOULD NOT contain a reference to the type definition `xsd:notation` or to a type derived from that type.

[Best Practice 8-4]

A SISO XML schema SHOULD NOT contain the element `xsd:notation`.

Rationale

XML schema notations allow the attachment of system and public identifiers on fields of data. The notation mechanism does not play a part in validation of instances.

8.1.3 XML Schema Inclusion

[Best Practice 8-5]

XML schemata that do not have namespaces SHOULD NOT be included.

Rationale

Element `xsd:include` brings XML schemas defined in separate files into the current namespace. Inclusion of XML schemas that do not have namespaces also complicates XML schema
understanding. This inclusion makes it difficult to find the realization of a specific XML schema artifact and create aliases for XML schema components that should be reused.

8.1.4 No XML Schema Redefinition

[Best Practice 8-6]
The XML schema SHOULD NOT contain the element `xsd:redefine`.

Rationale
The `xsd:redefine` element allows an XML schema document to restrict and extend components from a namespace, in that very namespace. Such redefinition introduces duplication of definitions, allowing multiple definitions to exist for components from a single namespace.

8.1.5 Wildcard Restrictions
There are many constructs within XML schema that act as wildcards. That is, they introduce buckets that may carry arbitrary or otherwise non-validated content. Such constructs violate [Principle 8], and as such provide implicit workarounds for the difficult task of agreeing on the content of data models. Such workarounds should be made explicitly, outside the core data model.

8.1.5.1 No Unconstrained Type Substitution

[Best Practice 8-7]
The XML schema SHOULD NOT reference the type `xsd:anyType`.

Rationale
XML Schema has the concept of the "ur-type," a type that is the root of all other types. This type is realized in XML schemas as `xsd:anyType`.

SISO XML schemas should not use `xsd:anyType`, because this feature permits the introduction of arbitrary content (i.e., untyped and unconstrained data) into an XML instance. SISO intends that the XML schemas describing that instance describe all constructs within the instance.

8.1.5.2 No Unconstrained Text Substitution

[Best Practice 8-8]
The XML schema SHOULD NOT reference the type `xsd:anySimpleType`.

Rationale
XML Schema provides a restriction of the "ur-type," which contains only simple content. This provides a wildcard for arbitrary text. It is realized in XML schema as `xsd:anySimpleType`.

SISO XML schemas should not use `xsd:anySimpleType` because this feature is insufficiently constrained to provide a meaningful starting point for content definitions. Instead, content should be based on one of the more specifically defined simple types defined by XML Schema.

8.1.5.3 Untyped Elements Should Be Abstract

[Best Practice 8-9]
Within the XML schema, an element declaration with the attribute `name` and without the attribute `type` SHOULD carry the attribute `abstract` with the value `true`.

Rationale
Untyped element declarations act as wildcards that may carry arbitrary data. By declaring such types abstract, this XSG supports the creation of type-independent semantics without allowing arbitrary content to appear in XML instances.
8.1.5.4 No Untyped Attributes
[Best Practice 8-10]
Within the XML schema, an attribute declaration with attribute name SHOULD carry the attribute type.

Rationale
Untyped XML schema attributes allow arbitrary content, with no semantics. Attributes should have a type so that specific syntax and semantics will be provided.

8.1.5.5 No Unconstrained Attribute Substitution
[Best Practice 8-11]
The XML schema SHOULD NOT contain the element xsd:anyAttribute.

Rationale
The xsd:anyAttribute element provides a wildcard, where arbitrary attributes may appear.

8.1.6 Component Naming Restrictions
All XML components must be named. That is, type definitions, and element and attribute declarations must be given explicit names — local and anonymous component definition is discouraged. Note that XML Schema enforces the placement of attribute group and model group definitions as top-level components, which forces the components to be named.

8.1.6.1 No Anonymous Type Definitions
[Best Practice 8-12]
Within the XML schema, any occurrence of the element xsd:complexType or xsd:simpleType SHOULD appear as an immediate child of the element xsd:schema.

Rationale
SISO XML schemas should not include anonymous types. All XML schema "top-level" types (children of the document element) are required by XML Schema to be named. Making named XML type definitions top level ensures that they are globally reusable.

8.1.6.2 No Local Element Declarations
[Best Practice 8-13]
Within the XML schema, any element declaration carrying the attribute name SHOULD appear as an immediate child of the document element xsd:schema.

Rationale
All XML schema components defined by SISO XML schemas should be named, accessible from outside the defining XML schema, and reusable across XML schemas. Local element definitions provide named elements that are not reusable outside the element’s defined context. Making named XML elements top level ensures that they are globally reusable.

8.1.6.3 No Local Attribute Definitions
[Best Practice 8-14]
Within the XML schema, any attribute declaration owning the attribute name SHOULD appear as an immediate child of the document element xsd:schema.

Rationale
All SISO XML schema components are named, accessible from outside the defining XML schema, and reusable across XML schemas. Local attribute definitions provide named attributes that are not
reusable outside the attribute’s defined context. Making named XML attributes top level ensures that they are globally reusable.

8.1.7 No Uniqueness Constraints

[Best Practice 8-15]
The XML schema SHOULD NOT contain any of the elements xsd:unique, xsd:key, xsd:keyref, xsd:selector, or xsd:field.

Rationale
XML Schema provides XML product development groups with the ability to apply uniqueness constraints to XML Schema validated content. These mechanisms, however, establish relationships in a way that is very difficult to understand, extend, and keep consistent through XML schema reuse.

8.1.8 Model Group Restrictions

Complex content definitions in XML Schema use model group XML schema components. These XML schema components, xsd:all, xsd:choice and xsd:sequence, also called compositors, provide for ordering and selection of particles within a model group.

XML Schema defines a particle as an occurrence of xsd:element, xsd:sequence, xsd:choice, xsd:any (wildcard) and xsd:group (model group) within a content model. For example, an xsd:sequence within an XML schema complex type is a particle. An xsd:element occurring within an xsd:sequence is also a particle.

8.1.8.1 No Recursively Defined Model Groups

[Best Practice 8-16]
Within the XML schema, any immediate child of a model group xsd:sequence element SHOULD be one of xsd:annotation, xsd:element, xsd:choice, xsd:sequence.

[Best Practice 8-17]
Within the XML schema, any immediate child of a model group xsd:choice element SHOULD be one of xsd:annotation, xsd:sequence or xsd:element.
Note: This best practice precludes the use of the particle xsd:group, xsd:any as immediate children of a model group.

Rationale
XML Schema provides the capability for model groups to be recursively defined, i.e. a sequence may contain a sequence, and a choice may contain a choice. These best practices are designed to keep content models simple, comprehensive, and reusable. The content of an element should boil down to a simple list of elements, defined in as straightforward a manner as is possible to meet requirements.

8.1.8.2 XML Schema Particle Cardinality Restrictions

[Best Practice 8-18]
Within the XML schema, if the element xsd:sequence carries the attribute minOccurs, it SHOULD set the value for the attribute to 1.

[Best Practice 8-19]
Within the XML schema, if the element xsd:sequence carries the attribute maxOccurs, it SHOULD set the value of the attribute to 1.

Rationale
XML Schema allows each particle to specify cardinality (how many times the particle may appear in an instance). This XSG recommends restricting the cardinality of xsd:sequence particles to exactly one, to ensure that content model definitions are defined in as straightforward a manner as possible.
Discussion
Note that element declarations acting as a particle (particles formed by xsd:element) may have any cardinality; they are not constrained by this best practice. Should a user desire the behavior that would be obtained from the use of special cardinalities on these particles, the developer should define them within explicitly named elements.

8.1.8.3 Default Value Restrictions
XML Schema provides the capability for element and attribute declarations to provide default values when XML instances using those components do not provide values.

[Best Practice 8-20]
Within the schema, any element xsd:element SHOULD NOT carry the attribute default.

[Best Practice 8-21]
Within the schema, any element xsd:attribute SHOULD NOT carry the attribute default.

Rationale
The use of default values means that the act of validating a schema will insert a value into an XML instance where none existed prior to schema validation. Schema validation is for rejection of invalid instances, not for modifying instance content.

8.1.9 Block Substitution Restrictions
XML Schema provides a mechanism that will prevent substitution for an element declaration or type definition. That is, an element declaration may declare one or more of the following:

1. An instance of this element declaration may not substitute an extended type.
2. An instance of this element declaration may not substitute a restricted type.
3. An instance of this element declaration may not substitute another element.

These restriction mechanisms are very useful in instances; they allow restriction of content models down to exact types and elements. However, in shared data models, they limit reuse and customization options, in opposition to [Principle 11].

[Best Practice 8-22]
Within the XML schema, an element declaration SHOULD NOT carry the attribute block.

[Best Practice 8-23]
Within the XML schema, a complex type definition SHOULD NOT carry the attribute block.

[Best Practice 8-24]
Within the XML schema, the document element xsd:schema SHOULD NOT carry the attribute blockDefault.

Rationale
Restriction of substitution options reduces capacity for reuse. In particular, setting the block value at the XML schema level complicates understanding of component definitions.

8.1.10 Final Value Restrictions
XML Schema provides the capability for type definitions and elements to declare a final value. This value prevents the creation of derived components. In shared data models, this capability limits reuse and customization options, in opposition to [Principle 13].

[Best Practice 8-25]
Within the XML schema, a simple type definition SHOULD NOT carry the attribute final.
[Best Practice 8-26]  
Within the XML schema, a complex type definition SHOULD NOT carry the attribute final.

[Best Practice 8-27]  
Within the XML schema, an element declaration SHOULD NOT carry the attribute final.

[Best Practice 8-28]  
Within the XML schema, the document element xsd:schema SHOULD NOT carry the attribute finalDefault.

Rationale  
Restriction of derivation options reduces capacity for reuse. As well, the use of finalDefault complicates understanding of XML schemas.

8.2 XML Schema xsd:schema Document Element

The features of XML Schema allow for flexibility of use for many different and varied types of implementations. This XSG recommends consistent use of these features.

[Best Practice 8-29]  
Within the XML schema, the document element xsd:schema SHOULD carry the attribute targetNamespace.

[Best Practice 8-30]  
Within the XML schema, the value of the attribute targetNamespace on the document element xsd:schema SHOULD match the production <absolute-URI> as defined by [9].

Rationale  
XML schemas without defined namespaces provide definitions that are ambiguous, in that they are not universally identifiable.

Absolute URIs are the only universally meaningful URIs. URIs include both URLs and URNs. Finding the target namespace using standard XML Base technology is complicated and not specified by XML Schema. Relative URIs are not universally identifiable, as they are context-specific.

Discussion  
The document element xsd:schema may contain optional attributes attributeFormDefault and elementFormDefault. The values of these attributes are immaterial as each attribute defined by a SISO XML schema should be defined at the top level and so should be qualified with the target namespace of its declaration.

[Best Practice 8-31]  
Within the XML schema, the document element xsd:schema SHOULD carry the attribute version of the SISO Product the XML schema applies to.

[Best Practice 8-32]  
Within the XML schema, the value of the attribute version on the document element xsd:schema SHOULD follow the SISO product numbering schema as specified in SISO-ADM-001[1].

Rationale  
It is very useful to be able to tell one version of an XML schema from another. Apart from the use of namespaces for versioning, it is sometimes necessary to release multiple versions of XML schema documents. Such use might include:
• Error corrections or bug fixes
• Documentation changes
• Contact information updates

In such cases, a different value for the version attribute implies a different version of the XML schema. No specific meaning is assigned to specific version identifiers.

8.3 Namespace Imports

XML Schema requires that namespaces used in external references be imported using the xsd:import element. The xsd:import element appears as an immediate child of the xsd:schema element. An XML schema must import any namespace which

1. Is not the local namespace, and
2. Is referenced from the XML schema.

The behavior of import statements is not necessarily intuitive. In short, the import introduces a namespace into the XML schema containing the import; it has no transitive effect. If the namespaces of an import statement are not referenced from the XML schema, then the import statement has no effect. The import statement cannot be used to direct XML schema locations for XML schemas not referenced from the XML schema performing the import. The XML schema location directed by the import element may be overridden by user directive at the parser, or by being overridden by import elements from other XML schemas.

Imports of namespaces should be made as uniform as possible; all XML schemas in an XML schema set should agree on what XML schema location goes with a particular namespace. Otherwise, behavior may be dependent on the behavior of the parser and the order of components in instance documents.

8.3.1 xsd:import Element Restrictions

[Best Practice 8-33]

Within the XML schema, the element xsd:import SHOULD carry the attribute namespace.

[Best Practice 8-34]

Within the XML schema, the value of the attribute namespace owned by the element xsd:import SHOULD match the production <absolute-URI> as defined by [9].

Rationale

An import that does not specify a namespace is enabling reference to non-namespaced components. It is important that the namespace declared by an XML schema be universally defined and unambiguous. Use of the standard XML Base for processing is not specified by XML Schema, thus it is not supported here.

[Best Practice 8-35]

Within the XML schema, the element xsd:import SHOULD carry the attribute XML schemaLocation.

Rationale

An import that does not specify an XML schema location gives no clue to processing applications as to where to find an implementation of the namespace. Even though such a provided XML schema location may be overridden, it is important that an initial default be provided for processing.

[Best Practice 8-36]

Within the XML schema, the value of the attribute XML schemaLocation carried by the element xsd:import SHOULD match either the production <absolute-URI> or the definition of "relative-path reference," as defined by [9].
Rationale

The default value may be specified either as absolute or relative URIs. Since URNs are not resolvable, they are inappropriate for use in XML schemaLocation. The recommendation for using "relative-path reference" is necessary to avoid the more obscure syntax of "network-path reference" and the system-specific "absolute-path reference."

[Best Practice 8-37]

Within the XML schema, the value of the attribute XML schemaLocation carried by the element xsd:import SHOULD be resolvable to an XML schema document file that is valid according to [15] and [16].

Rationale

XML Schema requires that the object imported via xsd:import be an XML schema document. This best practice reinforces that requirement.

Discussion

Note that relative URI references are dereferenced from the location of the XML schema document performing the import, not from the location of an instance or other XML schema.

8.3.2 Including XML Content from Other Namespaces

Within an XML schema document, there are several mechanisms to include XML content that is not from the XML or XML schema namespaces. Those mechanisms are:

1. Carrying attributes from other than the XML or XML schema namespaces on an element in the XML schema namespace.
   By the best practices of XML Schema, any element may have attributes that are from other namespaces. These attributes do not participate in validation but may carry information useful to tools that process XML schemas.

2. Adding content to the elements xsd:appinfo and xsd:documentation.
   XML Schema allows arbitrary XML content to be included within annotations. Such XML does not participate in validation but may communicate useful information to XML schema readers or processors.

This XSG recommends all such XML content to be “schema-valid.” That is, XML content should have an XML schema, and it should validate against that XML schema. The XML schemas should be introduced via xsd:import elements within the XML schema that uses the imported content. This is for two reasons:

1. Some tools require imports of namespaces used within XML schemas and validate against those XML schemas.
2. The definition and the validity of content within XML schemas should be clear.

[Best Practice 8-38]

Within the XML schema, when a namespace other than the XML namespace or the XML schema namespace is used, it SHOULD be imported into the XML schema using the xsd:import element.

Rationale

This best practice ensures that used namespaces have recognizable defining sources and that they will cooperate with existing tools.

[Best Practice 8-39]

Within the XML schema, when a namespace other than the XML namespace or the XML schema namespace is used, its content SHOULD be valid with respect to the XML schema imported for that namespace.
Rationale

XML Schema does not address the XML Schema validity of content used for annotations or attributes on XML schema components. This best practice ensures that content used in such a manner is XML Schema valid. This encourages interoperable data definitions and XML schema documents.

8.4 Annotations

Annotations in XML Schema “provide for human and machine-targeted annotations of XML schema components.” [16] The two types: human-targeted and machine-targeted, are kept separate by the use of two separate container elements defined by XML schema: xsd:documentation and xsd:appinfo, respectively. Annotations document XML components for humans, in natural language, and machines in structured data. XML annotations alone are not a substitute for natural language specifications that define how an XML schema is used to further interoperability and reuse.

[Best Practice 8-40]

Within the XML schema, an element SHOULD have at most one instance of an element xsd:annotation as an immediate child.

Rationale

XML Schema allows annotations to be added to components in a fairly loose manner: there may be multiple annotations, that each have multiple documentation or appinfo elements. This flexibility in the syntax provides no additional expressivity but does complicate processing, so it should be avoided in XML schemas.

8.4.1 Human-Readable (Natural Language) Documentation

XML Schema describes the content of xsd:documentation elements as “user information.” This information is targeted for reading by humans in natural language (English). XML Schema does not say what form human-targeted information should take. Within SISO, user information should be plain text with no formatting or XML structure.

[Best Practice 8-41]

Within the XML schema, the content of the xsd:documentation element that constitutes the data definition of a component SHOULD be character information items as specified by [12].

Rationale

According to XML Schema, the content of xsd:documentation elements is intended for human consumption, whereas other structured XML content is intended for machine consumption. Therefore, the xsd:documentation element SHOULD NOT contain structured XML data. As such, any XML content appearing within a documentation element is in the context of human-targeted examples and should be escaped using &lt; and &gt;.

XML comments are not XML schema constructs and are not specifically associated with any XML schema-based components. As such, comments are not considered semantically meaningful.

[Best Practice 8-42]

XML comments SHOULD not be used for persistent information about constructs within the XML schema.

Rationale

Since XML comments are not associated with any specific XML schema construct, there is no standard way to interpret comments. As such, comments should be reserved for internal use, and XML schema annotations should be preferred for meaningful information about components. This XSG specifically defines how information should be encapsulated in SISO XML schemas via xsd:annotation elements.
8.4.2 Machine-Readable Annotations

XML Schema provides special annotations for support of automatic processing. XML Schema provides the element `xsd:appinfo` to carry such content and does not specify what style of content it should carry. In this XSG, `xsd:appinfo` elements carry structured XML content.

[Best Practice 8-43]

Within the XML schema, any immediate child of an `xsd:appinfo` element SHOULD be an element information item or a comment information item.

**Rationale**

Application information elements are intended for automatic processing, so they should contain machine-oriented XML data.

[Best Practice 8-44]

Within the XML schema, any element that is an immediate child of an `xsd:appinfo` element SHOULD be in a namespace.

**Rationale**

Use of default namespace is allowed, but content has to have a real namespace, and namespaces should be declared. The XML namespaces specification includes the concept of content not in a namespace. Non-namespaced data run counter to the principle of distinctly identifiable data definitions.

[Best Practice 8-45]

Within the XML schema, an element in the XML schema namespace SHOULD NOT occur as a descendent of any element `xsd:appinfo`.

**Rationale**

SISO XML schemas are designed to be processed very easily. Although uses of XML schema elements as content of `xsd:appinfo` elements could be contrived, it is not current practice and could seriously complicate the authoring of XML schema validators and processors, such as XSLT [17], which may evaluate XML elements by their namespaces and names. Avoiding the use of XML schema elements outside valid uses of XML schema will simplify such processing.

8.5 Comments

This XSG reserves comments for use in documenting meta information about the XML schema as a SISO product. Comments are used for including information such as the appropriate SISO copyright that applies to the product (reference SISO-ADM-005) [2]. Other uses include identifying the change request that resulted in a change to the XML schema product.

[Best Practice 8-46]

Comments should not be used to document the normative or descriptive aspects for any component of XML schemas.

8.6 Type Definitions

XML Schema provides a variety of ways to define new types. This section covers restrictions on defining complex types, with both simple and complex content.

8.6.1 Complex Type Definitions

XML Schema provides a large amount of flexibility in the creation of complex types. This XSG narrows the XML schema capability to a smaller set of constructs.
Note that best practices on discouraged constructs (Section 8.1.6.1: No Anonymous Type Definitions, above) preclude defining complex types as local types. All complex type definitions should be top-level, named components.

XML Schema makes a distinction between complex types with simple content versus complex types with complex content. Complex types with simple content (CSCs) have content that is not allowed to contain XML elements. Complex types with complex content (CCCs) have content that does contain XML elements. Since mixed content is discouraged in XML schemas by [Best practice 8-1], all complex types are either CSCs or CCCs.

### 8.6.2 Simple Content (CSC) Restrictions

Within a SISO XML schema, a complex type with simple content (CSC) can be created in one of two ways:

1. By extension of an existing CSC.
2. By extension of an existing simple type.

Both of these methods use the element `xsd:extension`.

[Best Practice 8-47]

Within the XML schema, the element `xsd:simpleContent` SHOULD have as an immediate child the element `xsd:extension`.

**Rationale**

This best practice ensures that the definition of a CSC will use the XML schema extension facility. This allows for the above cases while disallowing much more complicated syntactic options available in XML schemas.

Note that the applicability of the above best practice allows for use of `xsd:restriction` within `xsd:simpleContent`.

Although the two above methods have similar syntax, there are subtle differences. Adherence with best practices ensure that any complex type has the necessary attributes for representing IDs, metadata, and link metadata. So, case 1 does not require adding these attributes, as they are guaranteed to occur in the base type.

However, in case 2, where a new complex type is created from a simple type, the attributes for complex types must be added.

[Best Practice 8-48]

Within the schema, given an element `xsd:simpleContent` with a child `xsd:extension` owning an attribute `base`, if the attribute `base` has a value that resolves to the name of a simple type, then the element `xsd:extension` SHOULD have an immediate child element `xsd:attributeGroup`.

**Rationale**

This best practice ensures that a CSC that is created as an immediate extension of a simple type adds the attributes required for a complex type.

### 8.6.3 XML Schema Complex Content (CCC) Restrictions

[Best Practice 8-49]

Within the XML schema, given an element `xsd:complexContent` with a child `xsd:restriction` owning an attribute `base`, the attribute `base` SHOULD have a value that resolves to the name of a complex type that is a defined component.

[Rationale]

This ensures that a CCC defined through restriction has well-defined semantics.
8.7 Additional Definitions and Declarations

XML Schema provides a variety of ways to declare and define elements and attributes.

8.7.1 Element Declarations

Within SISO XML schemas, elements may be declared as abstract. Element declarations must be at the top level, as best practices in other sections preclude the use of local elements. Elements may be defined without a type, but any element declaration that has no type should be declared abstract by [Best practice 8-12], which precludes anonymous type definitions.

Within an element declaration, the attributes fixed, nillable, and substitutionGroup may be used as per the XML Schema. The attribute form is irrelevant, as SISO XML schemas should not contain local element definitions by [Best practice 8-13].

Element uses (element declarations acting as particles) should reference top-level named elements in an element use. This XSG allows any values for the XML schema properties maxOccurs and minOccurs.

Based on a variety of user requirements, all elements in the XML schemas are defined to allow a nil value. For example, the following XML instances are permitted in instances:

\[
<\text{nc:ActivityDate}></\text{nc:ActivityDate}>
\]

Or

\[
<\text{nc:ActivityDate}/>
\]

Nil value allowance or restriction is only significant to elements of nontextual types (e.g., dates and numeric values) and elements of text types that have restricted value space (e.g., code). This is because an unrestricted text typed element always contains the empty string (""") in its value space. However, for numeric values and restricted text type elements, this XSG allows users to tighten constraints as required in IEPDs by resetting nillable="false".

8.7.2 Attribute Declarations

Attribute declarations should be declared with a type by [Best practice 8-10], which precludes anonymous type definitions for attributes.

Within an attribute declaration, the attribute fixed may be used as per the XML Schema. Within an attribute declaration, the attribute form is irrelevant, as SISO XML schemas should not contain local attribute declarations.

Attribute uses (attribute declarations acting as particles) should be uses of top-level named attributes. SISO XML schemas should not define local named attributes within type definitions. Within an attribute use, the attributes fixed and use may be used as per this XSG.

9 XML SCHEMA MODELING BEST PRACTICES

This XSG provides a framework for modeling concepts and relationships as XML artifacts. The data model is implemented via XML Schema. However, XML Schema does not provide sufficient structure and constraint to enable translating from a conceptual model to an XML schema and then to instances of the concepts. This XSG provides additional guidance for modeling concepts.

9.1 xsd:schema Document Element Restrictions

[Best Practice 9-1]

Two XML schema documents should have the same value for attribute targetNamespace carried by the element xsd:schema, if and only if they represent the same set of components.
[Best Practice 9-2]

Two XML schema documents should have the same value for attribute `targetNamespace` carried by the element `xsd:schema`, and different values for attribute `version` carried by the element `xsd:schema` if and only if they are different views of the same set of components.

Rationale

These best practices embody the basic philosophy behind this XSG’s use of namespace components: a component is uniquely identified by its class (e.g. element, attribute, type), its namespace (a URI), and its name (an unqualified string). Any two matching component identifiers refer to the same component, even if the versions of the XML schemas containing each are different.

9.2 Annotations

SISO XML schemas define data models for the purpose of information exchange. A major part of defining data models is the proper definition of the contents of the model. What does a component mean, and what might it contain? How should it be used? SISO XML schemas contain the invariant part of the definitions for the data model. The set of definitions includes:

1. A text definition of each component: this describes what the component means. The term used in this specification for such a text definition is data definition.
2. The structural definition of each component: this is made up of XML schema component definitions, along with certain application information (appinfo).

When possible, meaning is expressed via XML schema mechanisms: type derivation, element substitution, specific types and structures, as well as names that are trivially parseable. Beyond that, SISO XML-specific syntax should be used, as discussed in this section.

9.2.1 Human-Readable (Natural Language) Documentation

By other best practices, an XML schema component should contain at most one `xsd:annotation` element. An element `xsd:annotation`, in turn, contains at most elements `xsd:documentation` and `xsd:appinfo`. The content of the first element `xsd:documentation` on a component is the data definition for the component.

[Best Practice 9-3]

Within the XML schema, any element `xsd:complexType` SHOULD be a documented component.

[Best Practice 9-4]

Within the XML schema, any element `xsd:simpleType` SHOULD be a documented component.

[Best Practice 9-5]

Within the XML schema, any element `xsd:element` that is an immediate child of an element `xsd:schema` SHOULD be a documented component.

[Best Practice 9-6]

Within the XML schema, any element `xsd:attribute` that is an immediate child of an element `xsd:schema` SHOULD be a documented component.

[Best Practice 9-7]

Within the XML schema, any element `xsd:enumeration` SHOULD be a documented component.

[Best Practice 9-8]

Rationale

Data definitions should be concise, precise, and unambiguous without embedding additional definitions of data elements that have already been defined once elsewhere (such as object classes). W3C XML Schema [4] states that definitions should not be nested inside other definitions. Furthermore, a data dictionary is not a language dictionary. It is acceptable to reuse terms (object class, property term, and qualifier terms) from a component name within its corresponding definition to enhance clarity, as long as the requirements and recommendations of [4] are not violated. This further enhances brevity and precision.

9.3 Simple Type Definitions

SISO places very few restrictions on the definition of simple types in XML schemas. The use of lists should be reserved for cases where the data are fairly uniform.

[Best Practice 9-10]

Within the XML schema, a simple type definition that uses xsd:list SHOULD NOT be defined if any member of the list requires a property or metadata that is different than other members of the list. All members of the list SHOULD have the same metadata and should be related via the same properties.

Rationale

The members of a list are not individually addressable by SISO metadata techniques. The members are also not individually addressable by properties; a property has a value of all the members of the list. SISO XML Guidance provides no method for individually addressing a member of a list. If an individual member of a list needs to be marked up in a manner different than other members of the list, the use of individual elements may be preferred to the definition of a list simple type.

9.4 Component Usage

[Best Practice 9-11]

Any type definition referenced by a component within the XML schema SHOULD be from one of the following:

1. The XML schema being defined.
2. An imported namespace.
3. The XML schema namespace.

Rationale

SISO XML schemas are based on other SISO XML schemas and the supporting namespaces. This simplifies processing and understanding of data.

[Best Practice 9-12]

Any element declaration referenced by a component within the XML schema SHOULD be from one of the following:

1. The XML schema being defined.
2. An imported namespace.
3. An external namespace, in accordance with the best practices for external XML schemas as specified by this specification.

[Best Practice 9-13]

Any attribute declaration referenced by a component within the XML schema SHOULD be from one of the following:
1. The XML schema being defined.
2. An imported namespace.
3. The XML namespace.
4. An external namespace, in accordance with the best practices for external XML schemas as specified by this specification.

**Rationale**

SISO XML schemas are based on other SISO XML schemas. All attributes and elements should be from SISO XML schemas, the XML namespace, or an external namespace. This applies to elements referenced for substitution groups as well. It does not apply to content of the XML schema (e.g., within annotations) or to the XML schema declarations themselves. It applies only to attributes and elements referenced by the XML schema components.

### 9.4.1 Reference Elements

In XML instances, relationships between data objects are expressed as XML elements:

1. Data objects are expressed as XML elements.
2. XML elements contain attributes and other elements.

In this way, there is generally some implicit relationship between the outer element (the "containing" element, also known as the parent element) and the inner elements (the contained elements, also known as the child elements). Such expression of relationships is said to be by containment.

Expression of all relationships via element containment is not always possible. Situations that cause problems include:

- **Circular relationships.** For example, suppose that object 1 has a relationship to object 2 and object 2 has a relationship to object 1. Expressed via containment, this relationship would result in infinite recursive descent.
- **Repeated relationships.** For example, suppose object 1 has a relationship to object 2 and object 3 has a relationship to object 2. Expressed via containment, this would result in a duplicate of object 2.

A method that solves this problem is the use of references. In C or assembler, a pointer would be used. In C++, a reference might be used. In Java, a reference value might be used. The method defined by the XML standard is the use of ID and IDREF. An IDREF refers to an ID. This XSG uses this method and assigns to it specific semantics.

**[Best Practice 9-14]**

Within the XML schema, a reference element and only a reference element SHOULD be defined to be of type IDREF.

**Rationale**

Reference elements should be of the IDREF type, and elements of the IDREF type should be reference elements. This best practice ensures that users always create reference elements using IDREF and cannot use IDREF for any other purpose.

**[Best Practice 9-15]**

Within the XML schema, any two elements of the form NCName and NCNameReference where the string value of NCName is the same in both forms, SHOULD be defined to have identical semantics. SISO XML guidance recognizes no difference in meaning between a reference element and an element that is not a reference element.

**Rationale**

Data instances may use concrete data elements and reference elements as needed, to represent the meaning of the fundamental data. There is no difference in meaning between reference and concrete...
data representations. The two different methods are available for ease of representation. No difference in meaning should be implied by the use of one method or the other.

Assertions that indicate "included" data are intrinsic, while referenced data are extrinsic, are not valid and are not applicable to data instances and data definitions.

XML Schema requires the use of an attribute of type IDREF to define reference elements. According to the best practices of XML, such an attribute must contain a value that is represented by an attribute of type ID. In an instance, the targets of IDREFs are expected to be values of the attribute ID.

9.5 XML Schema Using External XML Schemas

There are a variety of commonly used standards that are represented in XML Schema. SISO XML schemas may reference components defined by external XML schemas.

SISO XML schemas may work with external XML schemas by creating external adapter types.

A single method is used to integrate external components into SISO XML schemas; XML types are constructed from the external components.

![Diagram of XML Schema Using External Components]

Components defined by external XML schemas are called external components. An XML type may use external components. The goal in this method is to preserve, as a single unit, a set of data that embodies a single concept from an external standard.

For example, an XML type may be created to represent a bibliographic reference from an external standard. Such an object may be composed of multiple elements and types from the external standard. These pieces are put together to form a single XML type. For example, an element representing an author, a book, and a publisher may be included in a single bibliographic entry. An XML type built from these components may be used as any other XML type. To construct such a component, a SISO XML schema should first import an external XML schema.

[Best Practice 9-16]

Within the XML schema, an element xsd:import that imports a namespace defined by an external XML schema SHOULD be a documented component.
Rationale
A SISO XML schema has well-known documentation points. Therefore, an XML schema that imports a namespace need not provide additional documentation. However, when an external XML schema is imported, appropriate documentation should be provided at the point of import because documentation associated with external XML schemas is undefined and variable.

[Best Practice 9-17]
Within the XML schema, an adapter type SHOULD be composed of only elements and attributes from an external standard.

Rationale
An adapter type should contain the information from an external standard to express a complete concept. This expression should be composed of content entirely from an external XML schema. Most likely, the external XML schema will be based on an external standard with its own legacy support.

In the case of an external expression that is in the form of model groups, attribute groups, or types, additional elements and type components may be created in an external XML schema, and the adapter type may use those components.

[Best Practice 9-18]
Within the XML schema, an element reference used in an adapter type definition SHOULD be a documented component.

[Best Practice 9-19]
Within the XML schema, an attribute reference used in an adapter type definition SHOULD be a documented component.

Rationale
In normal type definition, a reference to an attribute or element is a reference to a documented component. Within an adapter type, the references to the attributes and elements being adapted are references to undocumented components. These components should be documented to provide comprehensibility and interoperability. Since documentation made available by external XML schemas is undefined and variable, documentation of these components is recommended at their point of use.

[Best Practice 9-20]
Within the XML schema, an adapter type SHOULD NOT be extended or restricted.

Rationale
Adapter types are meant to stand alone; each type expresses a single concept from an external XML schema, and adapter types are maintained in separate XML schemas that only contain adapter types. In this way, processors may easily switch modes, processing internal content in one way, and external content in another.

[Best Practice 9-21]
The value of the targetNamespace attribute owned by the xsd:schema document element of the subset XML schema SHOULD be the same as the value of the targetNamespace attribute owned by the xsd:schema document element of the XML schema.

9.6 XML Schema Container Elements

Often an XML document may contain multiple items, e.g. Dates, POCs, Associations, and Keywords. SISO recommends the use of an XML element as a container class to support such potential multiples.
Within the resulting XML document based on the XML schema, a container element would be represented with repeating individual sub elements.

```xml
<Keywords>
  <Keyword ms:taxonomy="Functionality" ddms:value="LVC Environment" />
  <Keyword ms:taxonomy="Mission Space" ddms:value="Air Warfare" />
  <Keyword ms:taxonomy="Products" ddms:value="Force Allocation" />
  <Keyword ms:taxonomy="Battle space" ddms:value="Mission" />
  <Keyword ms:taxonomy="Maturity" ddms:value="Mature - cyclical improvement" />
  <Keyword ms:taxonomy="Service" ddms:value="JCS" />
  <Keyword ms:taxonomy="Aggregation" ddms:value="Squadron" />
</Keywords>
```

Figure 4. Example of Container Element

In this example Keywords provides the container, which contains one or more Keyword subelements.

[Best Practice 9-22]

Within the XML schema, repeating elements of the same type SHOULD occur under a “containing” complex type.

Rationale

It is easier to support container classes in software if the containers follow a consistent pattern across XML schema implementations.

10 XML SCHEMA NAMING BEST PRACTICES

This section outlines the best practices used to create names for XML data components discussed in this document. Data component names should be understood easily both by humans and by machine processes. These best practices improve name consistency by restricting characters, terms, and syntax that could otherwise allow too much variety and potential ambiguity. These best practices also improve readability of names for humans, facilitate parsing of individual terms that compose names, and support various automated tasks associated with dictionary and controlled vocabulary maintenance.

10.1 Extension of XSD Namespace Simple Types

[Best Practice 10-1]

Within the XML schema, a complex type that is a direct extension of a simple type from the XML schema namespace simple type MAY use the same local name as the simple type if and only if the extension adds no content other than the attribute group structures:SimpleObjectAttributeGroup.

Rationale

It is useful to build complex type bases for further extension. Inducing name changes for NIEM-provided complex type bases would work against understanding by users already familiar with the names of the XML schema namespace simple types being extended.

10.2 Usage of English

[Best Practice 10-2]

The name of any XML schema component defined by the XML schema SHOULD be composed of words from the English language, using the Merriam-Webster's Collegiate Dictionary [8] (current edition), as specified by the SISO Policy for the Style and Format of SISO Documents (SISO-ADM-005) [2].
Rationale

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

10.3 Characters in Names

[Best Practice 10-3]

The name of any XML schema component defined by the XML schema SHOULD contain only the following characters:

- Upper-case letters (‘A’-‘Z’).
- Lower-case letters (‘a’-‘z’).
- Digits (‘0’-‘9’).
- Hyphen (‘-’).

Other characters, such as the underscore (‘_’) character and the period (‘.’) character SHOULD NOT appear in component names.

[Best Practice 10-4]

The hyphen character (‘-’) MAY appear in component names only when used as a separator between parts of a single word, phrase, or value, which would otherwise be incomprehensible without the use of a separator.

Rationale

Names of standards and specifications, in particular, tend to consist of series of discrete numbers. Such names require some explicit separator to keep the values from running together. The separator used within this XSG is the hyphen.

Names of XML components follow the best practices of XML schema, by [Best practice 8-3]. XML components should also follow the best practices specified for each type of XML schema component.

10.4 Character Case

[Best Practice 10-5]

Within the XML schema, any attribute declaration SHOULD have a name that begins with a lower-case letter (‘a’-‘z’).

[Best Practice 10-6]

Within the XML schema, any XML schema component other than an attribute declaration SHOULD have a name that begins with an upper-case letter (‘A’-‘Z’).

Camel case is the practice of writing compound words or phrases where the words are joined without spaces and are capitalized within the compound words [10].

[Best Practice 10-7]

The name of any XML schema component defined by the XML schema SHOULD use the camel case formatting convention.

Rationale

These best practices establish lower camel case for all SISO components that are XML attributes and upper camel case for all SISO components that are types, elements, or groups.
10.5 Character Sets

[Best Practice 10-8]
Each instance XML document to include schemas SHOULD utilize UTF-8 encodings.

Rationale
All XML processors must be able to read entities in both the UTF-8 and UTF-16 encodings as required by [12]. The best practice of using UTF-8 encodings for instance documents only applies to SISO, and not to outside stakeholders who wish to use SISO standards in simulation architectures that employ other character sets and languages.

10.6 Use of Acronyms and Abbreviations

Acronyms and abbreviations have the ability to improve readability and comprehensibility of large, complex, or frequently used terms. They also obscure meaning and impair understanding when their definitions are not clear or when they are used injudiciously. Acronyms and abbreviations should be used with great care. Acronyms and abbreviations that are used should be documented in the natural language specification of the XML schema product, and used consistently throughout the XML schema.

[Best Practice 10-9]
The XML schema SHOULD consistently use acronyms, abbreviations, and word truncations within defined names.

[Best Practice 10-10]
Acronyms, abbreviations SHOULD be defined in the natural language specification of the XML schema product, and used consistently throughout the XML schema.

11 INSTANCE (SAMPLE) XML DOCUMENTS

Sample XML documents have the ability to improve the comprehensibility of applying XML schemas, as intended, to construct XML documents. They also obscure meaning and impair understanding when their implementations do not utilize all documented components of an XML schema.

[Best Practice 11-1]
Every XML schema product SHOULD include a sample (instance) XML document.

[Best Practice 11-2]
Every sample XML document accompanying an XML schema product should include at least one occurrence of every documented component defined in the XML schema.

Rationale
Sample SISO XML documents improve comprehension of an XML schema product by depicting how XML documents are intended to be constructed in XML.

12 DEPLOYING SISO XML DOCUMENTS

SISO encourages the distribution and sharing of its XML schemas. SISO XML schemas are to be stored on http://www.sisostds.org/schemas/ within the root folder with no directory hierarchy. Alternatively, the XML schemas can be stored on other sites to include open source organizations. An index.htm (catalog) file is also maintained to show the available XML schemas and supporting examples, independent of where the XML schema products are physically stored. The index.htm file specifies the following information for each XML schema product:

- Target Namespace – The target namespace of the XML schema product.
- Version – The product version of the XML schema product.
- XML Schema Location – The web accessible location URL of the XML schema product.
• SISO Product Number – The assigned SISO product number of the XML schema product.