

# **Simulation Interoperability Standards Organization (SISO)**

## **Guidelines for: Military Scenario Definition Language (MSDL)**

**X April 2007**

**Prepared by:**

**Simulation Interoperability Standards Organization**

**Military Scenario Definition Language  
Drafting Group**

Copyright © 2007 by the Simulation Interoperability Standards Organization (SISO), Inc.

P.O. Box 781238

Orlando, FL 32878-1238, USA

All rights reserved.

Permission is hereby granted for SISO developing committee participants to reproduce this document for purposes of SISO product development activities only. Prior to submitting this document to another standards development organization for standardization activities, permission must first be obtained from the SISO Standards Activity Committee (SAC). Other entities seeking permission to reproduce this document, in whole or in part, must obtain permission from the SISO Executive Committee (EXCOM).

SISO EXCOM

P.O. Box 781238

Orlando, FL 32878-1238, USA

DRAFT

## Table of Contents

<b>1</b>	<b>SCOPE.....</b>	<b>1</b>
1.1	IDENTIFICATION .....	1
1.2	DOCUMENT OVERVIEW .....	1
<b>2</b>	<b>REFERENCE DOCUMENTS .....</b>	<b>1</b>
2.1	SISO DOCUMENTS .....	1
2.2	TECHNICAL DOCUMENTS .....	1
2.3	MSDL PDG DOCUMENTS .....	1
<b>3</b>	<b>STYLE AND BEST PRACTICES.....</b>	<b>1</b>
3.1	STYLE .....	1
3.1.1	<i>Case</i> .....	1
3.1.2	<i>Elements</i> .....	2
3.1.3	<i>Attributes</i> .....	2
3.1.4	<i>Types</i> .....	2
3.1.5	<i>Modality</i> .....	3
3.1.6	<i>Keys and Key References</i> .....	3
3.2	BEST PRACTICES .....	3
3.2.1	<i>Use of Subsets</i> .....	3
3.2.2	<i>References to External Standards</i> .....	4
3.2.3	<i>Object Representation</i> .....	4
3.2.4	<i>Object Referencing</i> .....	4
3.2.5	<i>Element Scope</i> .....	5
3.3	DOCUMENTATION .....	5
3.3.1	<i>XML Spy Generated Documentation</i> .....	5

## Table of Tables

Table 1:	Simple Type Naming Convesion.....	2
----------	-----------------------------------	---

# Guidelines for: Military Scenario Definition Language (MSDL)

## Initial Draft

## Change History

[illegible]

# 1 Scope

## 1.1 Identification

The XML Coding Standards are followed in the development of MSDL schema. This applies to both candidate and selected extensions to the standard.

## 1.2 Document Overview

This document specifies the coding practices to be followed in specifying MSDL XML schema.

# 2 Reference Documents

## 2.1 SISO Documents

SISO-ADM-002-2006 SISO Policies & Procedures

SISO-ADM-003-2005 SISO Balloted Products Development Process

## 2.2 Technical Documents

- [Extensible Markup Language \(XML\)](#), Word Wide Web Consortium

## 2.3 MSDL PDG Documents

- [MSDL Product Development Plan \(PDP\)](#)

# 3 Style and Best Practices

The following sections outline best practices in XML style and standards to be followed.

## 3.1 Style

Style specifies general approach to specifying discrete elements of MSDL schema.

### 3.1.1 Case

The case of MSDL elements is specified in Proper Case. The specific format of case varies dependent on the context (enumerations, elements, attributes, types, etc.)

- Elements – The case of elements is specified in Proper Case, with no underscores in the element names. Acronyms don't reflect a preferred practice, but when used should be in all caps.
- Attributes – N/A. Attributes are not currently used in the formal MSDL standard.
- Enumerations – Enumerations are specified in upper case with underscores between words and acronyms.
- Types – Simple and complex types should follow the same guidelines as elements.

### 3.1.2 Elements

All elements should reference MSDL sub-elements, simple types, or coded enumerated types. All many associated restrictions as can be reasonably applied w/o falsely constraining the intended use shall be specified by MSDL data type. In all cases each XML element needs to include annotations of the elements.

#### 3.1.2.1 Types & Restrictions

Each element must include the specification of a data type to include associated restrictions that apply to the element (patterns, max/min values, enumerations, etc). In cases where MSDL provides for application specific data, such as is found in application options, the type should explicitly call out the xs:anyType.

#### 3.1.2.2 Annotations

Annotations should identify the standard from which an element was derived. See the section Candidate Extensions for details. All elements and types shall be annotated. Element references will be annotated when their documentation changes (is more specific) based on the use of the reference. For example an ObjectHandle reference shall be annotated in specific context to the object in which the handle is used.

#### 3.1.2.3 Global Elements

All elements shall be defined globally; unless the data type of a common element varies as it is used in the schema. In cases where restrictions vary by the location of an element, the element shall be defined locally to the higher element, with a reference to the MSDL restricted type.

### 3.1.3 Attributes

Attributes are not to be used in the MSDL standard. The intent is to enable candidate/evaluative extensions to MSDL to be specified as attributes for test and evaluation purposes w/o affecting the basic MSDL structure.

### 3.1.4 Types

All simple types shall be named to reflect the data type as follows:

Table 1: Simple Type Naming Convesion

Type	Name	Description
xs:string	text[name][length]	All strings defined within the MSDL namespace, without enumeration or pattern restrictions (length restriction alone).
xs:string	enum[name]	Enumerated restrictions defined within the MSDL namespace.
xs:string	pattern[name][length]	Pattern restrictions defined within the MSDL namespace.
xs:int	[u]integer[name][length]	All integer values where the Length is in

xs:integer xs:long		digits. The “u” is included for unsigned integers.
xs:float	float[name][d_f]	All floating point values where d_f is the number of total digits (d) and the number of fractional digits (f).
xs:boolean	boolean[name]	All boolean values (true, false).

59

60 This approach to naming the simple types ensures the types reflect a functional  
61 representation, not their use. The use is specified by the elements, in how the simple  
62 types are *used*. The [name] reflects the name of the functional type.

### 63 **3.1.5 Modality**

#### 64 **3.1.5.1 All**

65 Modality of complex elements should be set to all for those elements which have more  
66 than a single child element. Exceptions shall be agreed to by the drafting group.

#### 67 **3.1.5.2 Choice**

68 Each choice element should include in its parent an enumerated element that declares the  
69 choice made. This provides for verification of the choice, and identifies the proper  
70 choice in cases where two choices are included.

#### 71 **3.1.5.3 Sequence**

72 Sequences are used for collections of like elements and topmost schema elements in  
73 order to provide for SAX functionality. Collections cannot share a parent element with  
74 any other element (the parent or collection can have only one child element).

### 75 **3.1.6 Keys and Key References**

## 76 **3.2 Best Practices**

77 Best practices identify how XML is used when specifying functional relationships  
78 internal to and external to in the MSDL standard.

### 79 **3.2.1 Use of Subsets**

#### 80 **3.2.1.1 Standard Schema**

81 MilitaryScenario.xsd – Defines the topmost MSDL element (MilitaryScenario), and  
82 includes all subordinate schemas making up the standard. These topmost elements shall  
83 be sequenced to ensure SAX compatibility, where the sequence (order) is in the  
84 dependency order that minimizes the number of passes required to consume MSDL.

85 msdlElements.xsd – Includes all MSDL elements beneath the top level MilitaryScenario  
86 element for the approved/selected schema content.

msdlSimpleTypes.xsd - Includes all MSDL simple types. All elements will reference a simple type (no simple types will be defined locally to an MSDL schema element).  
msdlComplexTypes.xsd – Includes all MSDL complex types. Complex types are defined when specific MSDL XML structures are used repeatedly in the schema.

### **3.2.1.2 Candidate Extensions**

Candidate extensions should be defined in a standalone schema subset that is then included in the MilitaryScenario schema subset. As an objective, candidate element extensions should not be included or otherwise injected directly with elements in the msdlElements subset. This policy helps ensure candidate extensions are cleanly decoupled from the current standard. Where problems exist in context to existing content (extensions are called for), attributes should be used to reflect those extensions.

### **3.2.2 References to External Standards**

If a reference is to be included in the schema, annotations, not element names should be used to reflect the standards from which they were derived. The version of the standard should not be specified directly in the schema, but must be included in the annotations and the specification document. Specific guidelines regarding references to external specifications include:

- The identification of a standard includes its version.
- External XML specification shall be imported (not included) globally within the MSDL spec as an XML subset. The namespace shall be qualified and comments shall be required to include links to the external standard's web pages.
- Elements and types shall be properly annotated even if a reference to another standard is provided.
- A reference to an external standard shall only appear in the XML schema or if the concepts and enumerations of that standard are used unchanged. If there are a few interpretation issues, the reference should not appear in the XML schema, but shall appear in the specification documentation. Any interpretation details of these standards shall appear in the specification document, but not the schema.

### **3.2.3 Object Representation**

All objects should be defined as a complex type and include the global element ObjectHandle as the unique identifier of the object.

### **3.2.4 Object Referencing**

References between MSDL elements shall use ObjectHandles as the keys being referenced. Because there is not standard convention for specifying keys and key references across XML implementations, the schema restrictions on keys and keyRefs is not mandatory.



#### **3.2.4.1 Uni-Directional References**

References to other MSDL objects are to occur based on each object's ObjectHandle. The convention for naming these references is to name the element by the referenced element name with the word "Handle" appended. For example a UnitHandle references a Unit.ObjectHandle. Modifiers (adjectives) can be included for clarity. For example, UnitOwnerHandle would indicate an object is owned by a unit object.

#### **3.2.4.2 Bi-Directional References**

It is not recommended that truly circular references between child elements of two (or one) instanced complex elements be included in MSDL. In this context "truly" means two instanced objects reference each other circularly. Circular references should be instantiated by the application.

Note: The standard may include circular references in schema, as long as guidelines mitigate the risk of a truly circular relationship. For example a unit has to reference a superior unit. But no unit can be its own superior, so the risk is mitigated.

### **3.2.5 Element Scope**

In general it is not necessary to define a global type/ element when the type/element is a default native XML type or is unique to the MSDL schema.

#### **3.2.5.1 MSDL Types**

Define global data types uniquely based on how they vary from the base native XML type used.

#### **3.2.5.2 MSDL Elements**

Define global types for elements. No elements should be assumed to be unique, this will minimize changes when an element needs to be reused in context to future extensions.

## **3.3 Documentation**

### **3.3.1 XML Spy Generated Documentation**

For common convention, all documentation on MSDL should use XML spy generated figures.