SISO-STD-019-2020

Standard for Command and Control Systems - Simulation Systems Interoperation

Version 1.0

25 April 2020

Prepared by:
Command and Control Systems - Simulation Systems Interoperation
Product Development Group

P.O. Box 781238
Orlando, FL 32878-1238, USA

SISO Inc. Board of Directors
P.O. Box 781238
Orlando, FL 32878-1238, USA

Revision History

Version	Section	Date (MM/DD/YYY)	Description
1.0	All	29 March 2020	Initial version from balloting

Participants

At the time this product was submitted to the Standards Activity Committee (SAC) for approval, the Command and Control Systems - Simulation Systems Interoperation Product Development Group had the following membership and was assigned the following SAC Technical Area Director:

Product Development Group

Dr. J. Mark Pullen (Co-Chair)
Kevin Galvin (Co-Chair)
Bruno Gautreau (Vice-Chair)
Dr. Douglas Reece (Vice-Chair)
Dr. Rob Wittman (Vice-Chair)

Dr. Curtis Blais (SAC Technical Area Director)

Jeff Abbott
Elaine Blount
Donald Brutzman
Nicholas Clark
Douglas Corner
Anthony Cramp
Xavier Cuneo
Thomas DeCarlo
Todd Decosta
Bradford Dillman
Uwe Dobrindt
Bruno Gautreau
Arno Gerretsen
Kevin Gupton
Paul Gustavson
Mark Hazen
Frank Hill
Elizabeth Hosand
Lionel Khimeche
Patrice Le Leydour
Kenneth LeSueur
Sean Litton
Sebastien Loze
Lance Marrou
Craig Marsden
Priscilla McAndrews
Mark McCall
Ole Martin Mevassvik
Michael Montgomery
Laurent Mounet
William Oates
Laurent Prignac
Nelson Reynolds
David Ronnfeldt
James Ruth
Roy Scrudder
Tommy Shook
John Shue
Robert Siegfried
Samuel Singapogu
Eric Whittington

The Product Development Group would like to especially acknowledge those individuals that significantly contributed to the preparation of this product as follows:

PDG Drafting Group

Dr. Samuel Singapogu (Lead Editor)
Thomas DeCarlo (Secretary)

Curtis Blais
Douglas Corner
Magdalena Dechand
Kevin Galvin
Bruno Gautreau
Lt Col Jens-Inge Hyndoy
Dr. Mark Pullen
Dr. Douglas Reece
James Ruth
Dr. Rob Wittman

The following individuals comprised the ballot group for this product.

Ballot Group

Deryck Arnold
Mohammad Ababneh
Curtis Blais
Davor Braut
Donald Brutzman
Bruce Clay
Douglas Corner
Anthony Cramp
Todd DeCosta
Saikou Diallo
Peter zu Drewer
Kevin Galvin
Bruno Gautreau
Jean-Louis Gougeat
Kevin Gupton
Elizabeth Hosang
Lionel Khimeche
Katherine Morse
Bharat Patel
J. Mark Pullen
Douglas Reece
Samuel Singapogu
Geir Sletten
Jeffrey Sugden

When the Standards Activity Committee approved this product on 04 June 2020, it had the following membership:

Standards Activity Committee

David Ronnfeldt (Chair)
Curtis Blais (Vice Chair)
Katherine Ruben (Secretary)

Grant Bailey
Brad Dillman
David Graham
Peggy Gravitz
John Hughes
Patrice Le Leydour
Sebastien Loze
Lance Marrou
Chris McGroarty
David ‘Fuzzy’ Wells

Executive Committee

Robert Lutz (Chair)
Jeff Abbott (Vice Chair)
Kenneth Konwin (Secretary)

Damon Curry
Paul Gustavson
Kurt Lessmann
Mark McCall
Lana McGlynn
Chris Metevier
Katherine Morse
David Ronnfeldt
Stefan Sandberg
Robert Siegfried

Introduction

Command and Control Systems to Simulation Systems Interoperation (C2SIM) is a standard for expressing and exchanging Command and Control (C2) information among C2 systems, simulation systems, and robotic and autonomous (RAS) systems in a coalition context.

Military operations in today's world are increasingly driven towards coalition participation and as such are dependent on effective interoperation among participating coalition systems. The growth of digitized C2 systems and the need for coalition interoperation has created a need for standards to represent and exchange digitized C2 information and for these systems to interoperate.

The Simulation Interoperability Standards Organization (SISO) supports the identification and creation of standards and guidelines to help in the interoperability of simulation and more broadly to tackle issues related to M&S. Two main types of documents exist: guidelines and standards, the first mostly related to processes definition and the second mostly related to structuring data, with or without a reference model associated.

Guidelines describe processes and their underlying concepts for specific purpose. Two existing guidelines are of interest in the frame of this document. First, the *Recommended Practice for Distributed Simulation Engineering and Execution Process*, (DSEEP) Institute of Electrical and Electronics Engineers (IEEE) Standard 1730TM-2010: this provides generic engineering guidelines for distributed simulation. Most of the time, C2SIM works fit within an engineering approach involving distributed simulation. The added value of using such a process is stated in the DSEEP introduction: "By providing abstract representations of highly complex systems along with the operational environment in which the system exists, users can gain an understanding of the performance and behavior of the system that would not be achievable otherwise." Second, the *Guideline on Scenario Development for Simulation Environments*, (GSD): C2SIM interoperability provides the capability of being able to command and control in a given simulated situation. The interest of using a format for specifying C2SIM situations is summarized in the GSD introduction: "Scenarios defined by the user of a simulation environment are paramount sources of requirements for the engineers faced with planning and setting up a simulation environment. As such, well-specified scenarios are of utmost importance. If such scenarios are missing, this may lead to various problems due to misunderstandings when talking about the objectives and scope of a simulation environment."

The general architecture assumed by C2SIM is shown as a system of systems in Figure 1. The elements of this architecture are;

a) C2 System(s) - networked software/hardware systems capable of composing orders, submitting those orders, accepting situational awareness reports, and conveying the contents of those reports externally; these may exchange C2 information among themselves using other standards, for example Multilateral Interoperability Programme (MIP) [15].

b) Simulation System(s) - networked software/hardware systems, capable of using physics and organizational principles to project likely outcome when objects in real or virtual worlds interact in response to direction from orders and emit report messages; these may exchange detailed simulation data among themselves using other standards, for example High Level Architecture (HLA) [8].

c) Robotic and Autonomous Systems – networked software/hardware systems, mobile or immobile, that are capable of operation with remote direction (e.g., tele-operated systems) or general direction (semi-autonomous or fully autonomous); these too might exchange data for robotic or autonomous operation among themselves or with C2 systems using other standards.

d) Server System(s) - networked platforms, multiple of which may cooperate in a distributed configuration, capable of receiving subscriptions from participating C2 and simulation systems, accepting orders and reports from those systems, storing the latest state of simulated objects, forwarding reports to subscribing systems, and responding to queries with latest state of the objects. The server has marshalling functions (initializing and synchronizing) and data distribution functions. Data distribution provides for distribution of C2SIM messages to groups of C2, simulation, and RAS systems and could be replaced by multicast networking.

e) Real-time database to hold current system state and capture it in logs for after-action review and playback.

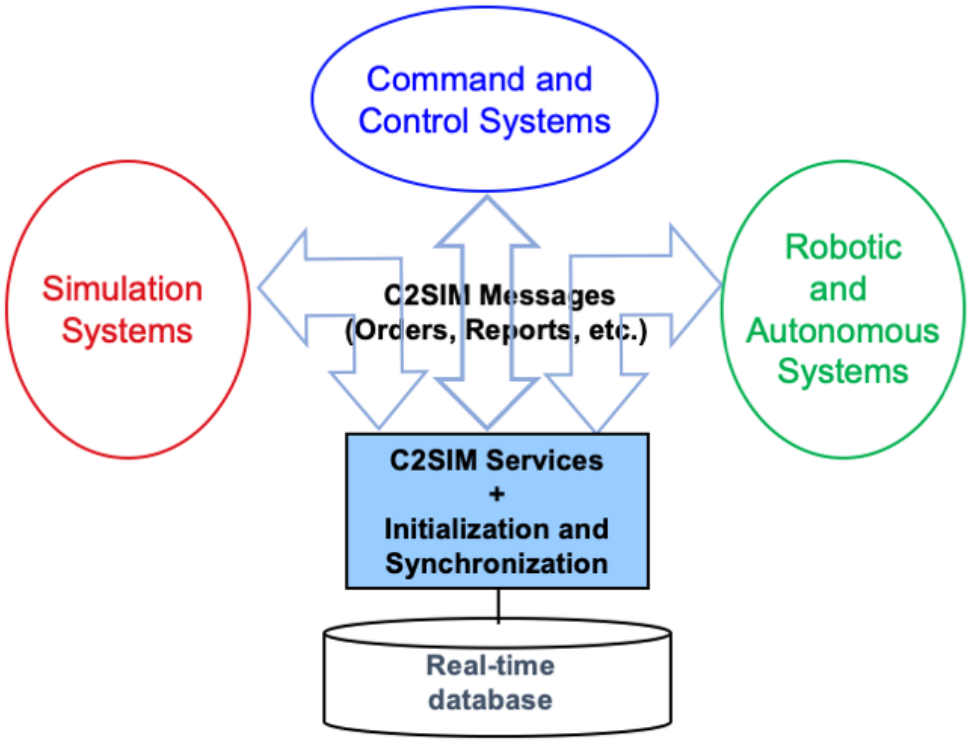

Figure 1: C2SIM General Architecture

Standards allow simulations and other applications to share a representation of reality. When discussing data structure and reference models, the first SISO standards that come in mind are the *Object Model Template* (OMT), IEEE 1516.2 and the *Real-time Platform Reference Federation Object Model* (RPR FOM), SISO-STD-001-2015. C2SIM focuses on structured tasking and reporting information interchange among a coalition of systems, independent from architecture and communication protocols. For example, in a Distributed Interactive Simulation (DIS) federation, C2SIM messages can be routed between federates separately from DIS packets or can be embedded in a DIS protocol data unit (PDU), such as embedding formatted messages in a Signal PDU. Similarly, with an HLA federation: C2SIM messages can be routed among federates separately from HLA interactions or can be embedded in HLA messages defined as interactions in a Federated Object Model. This is an architectural decision that is not part of the C2SIM standard. The focus goes to structuring data independent from architecture and communication protocols. Rather than having some number of different formats being sent across C2 systems, simulation systems, and RAS, C2SIM provides a single, unified format, with translations to those systems' internal formats at sending and receiving ends to and from C2SIM-formatted messages. Such translations become unnecessary when next-generation systems begin to become native C2SIM users. Use of C2SIM enables a single message format for message logging, information visualization, data processing, data querying, knowledge reasoning, and other purposes across the whole coalition of systems. C2SIM is the only international standard addressing information interchange across that breadth of systems. Its focus goes to structuring data independent from architecture and communication protocols.

Two other M&S previously standardized languages are expressed in a format independent from any communication protocols: the *Military Scenario Definition Language* (MSDL), SISO-STD-007-2008 and the *Coalition Battle Management Language* (C-BML), SISO-STD-011-2014. MSDL, used to support military scenario development, can be seen as the predecessor of C2SIM initialization; C-BML is used for exchanges of orders and reports (tasking and reporting). Tasking includes Orders and Requests; the two are essentially the same for purposes of this document. The MSDL and C-BML standards are the seedbed within which the current C2SIM was developed. Many references to these two standards are made thorough this document: a basic knowledge of these two standards is useful to have for the readers of this document.

MSDL and C-BML formed the first generation of standards for C2-simulation interoperation. However, in practice they were found to be difficult to combine and also insufficiently capable of extension, so the C2SIM Product Development Group (PDG) was formed by merging the PDGs of the two. The combined PDG has consolidated the functions of both C-BML and MSDL with the goal to create a unified standard that provides seamless integration among C2 initialization, tasking, and reporting with simulation systems and robotic systems. This unification endeavor goes well beyond simply merging the standards together: the scope, initially military-centric, is enlarged to all domains where C2 can exchange information with simulations. Structuring of data, formerly only available in the physical domain, i.e. Extensible Markup Language (XML) based language, is now available both in the logical domain (OWL ontology) and, in the physical domain XML Schema Definition (XSD) schema derived from the ontology. C2SIM defines physical or simulated activities based on a logical (platform-independent) data model, which is specified in a compatible set of ontologies: a Core that consists of data classes expected to be used by all (or almost all) interoperating C2 and simulation systems, a Standard Military Extension (SMX) integral to the basic standard that consists of additional classes, subclasses, and properties expected to be used by all (or almost all) interoperating military C2 and simulation systems, and a group of extensions associated with various domains (e.g. Ground Operations; Autonomous Systems). The C2SIM standard covers data structure mechanisms (format and syntax for C2SIM models) and rules for execution (systems initialization and exchange of messages among the systems). Communication procedures are adapted from the IEEE-adopted Foundation for Intelligent Physical Agents (FIPA) specification in order to ensure consistency of information flow.

The C2SIM PDG addressed these issues by combining the functions of MSDL and C-BML in a broad, general framework that provides for basic necessary functions:

- *Core and SMX Logical Data Models* provide, at a logical level (i.e., independent of how the data will be communicated), a core set of data elements common to most C2 and Simulation systems, combined with a standard way of adding to that core a collection of additional elements specific to a particular domain and/or context. The C2SIM standard document describes the process to serialize the Logical Data Model (LDM) classes, contained in a layered set of ontologies. This is transformed by a standard process into a physical data model that takes the form of an XML schema. Extension of C2SIM is achieved by adding a new ontology, compatible with the Core and with any other extensions with which the new extension is intended to comply. The data model includes elements for expressing and exchanging the following.
- *C2SIM Initialization* messages provide initial values for classes of the C2SIM Core and extension ontologies. This can be done by a preconfigured initialization file or by each participating system sending to a marshalling system (which might be the server) its initial values for those data classes indicated in the ontology as requiring initialization. The marshalling system consolidates the initial values and returns the aggregated initialization in a single document, upon receipt of a C2SIM Initialization message. It also provides for synchronization of systems in the C2SIM coalition by providing simulation control messages. C2SIM Initialization supersedes the MSDL version 1 standard, which is an XML message format developed with the purpose of initializing the operational environment (OE) in a wide variety of simulations and connected systems. Applications of the initialization messages include description of partial or complete start conditions for simulation execution (e.g., events and exercises) and contextual information defining the truth or belief conditions of actors in simulations. Other applications include defining simulation checkpoint (snapshots of past simulation conditions for reset or rollback operation), describing multiple courses of action (CoA), or contexts in the past, present, or future (e.g. planned, preset, anticipated, and objective states).

- *C2SIM Tasking-Reporting* defines a set of messages used by participating systems to distribute Order, Request and Report information to other participating systems, compliant with the C2SIM Core and extension ontologies. C2SIM Tasking-Reporting elements in the ontology will supersede the C-BML version 1 standard, which is an XML message format developed with the purpose of describing task and report messages in operational or simulation environments. The new product expands the potential range of tasking and situational awareness information relative to the C-BML v1 standard, though the expansion most often will come through extensions rather than adding to the C2SIM LDM Core. Task and report messages may be utilized during execution of simulations as runtime messages between real or simulated entities and as a common format for conveying information to and from tactical message formats based on the C2SIM LDM.
- *C2SIM-LandOperationsExtension* (LOX, a separate standard document, SISO-STD-020) will serve as an exemplar for other C2SIM extensions. It provides for continuity with military land operations aspects of MSDL and C-BML that are not included in the LDM Core and SMX, but are used by the international land military simulation community.

Table of Contents

List of Figures

List of Tables

1 Overview

Command and Control Systems to Simulation Systems Interoperation is being developed as a standard to support interoperation between C2 systems, simulation systems, and RAS, in a coalition context. The C2SIM PDG/PSG replaced the C-BML and MSDL PDGs and Product Support Groups (PSGs).

1.1 Scope

C2SIM covers the initialization, tasking, and reporting of forces. Initialization contains all information necessary for creating and describing forces, situation (weather, etc.), and control measures across interoperating C2 and simulation systems. Tasking and Reporting covers all information necessary to create tasks, provide situational reports, and manage forces between interoperating C2 system, simulation systems, and RAS. C2SIM deals with the exchanges of information among the types of systems listed above, to enable their collective initialization and operation as a coalition system of systems.

1.2 Purpose

This document describes the C2SIM standard and provides a normative description of the scope and use of C2SIM. The C2SIM standard is defined in a model driven framework that includes an ontology as formal LDM and a mechanism to extend the data model for specialized use. The normative C2SIM ontology is divided into two ontologies; Core C2SIM ontology and a Standard Military Extension (SMX) that imports the core C2SIM ontology.

1.3 Objectives

The primary objective of this standard is to provide developers of C2 systems, simulation systems, and RAS the ability to use the C2SIM LDM to generate an XML schema suitable to their needs to represent and exchange digitized C2 information. Using the derived XML schema, systems can generate data needed for scenario initialization, tasking, and reporting.

1.4 Intended Audience

The primary intended audience of this standard includes:

- C2 and simulation engineers developing interoperability among C2, simulation and robotic systems along with managing initialization and execution processes.
- Users and subject matter experts who define and specify requirements for C2 and simulation exchanges.

1.5 Introduction to Data Models

C2SIM has adopted a way to structure data based on three levels as commonly provided in data base management literature. Descriptions of the three levels are provided below. These are largely inspired from the ANSI-SPARC Architecture [16], an abstract design standard for a Database Management Systems, and other public sources:

- Conceptual: this model is used to explore high-level business/operational structures and concepts from a project stakeholder's point of view; it is built on a set of entity types and their relations, with a format that ranges from free text to structured class-relation diagrams. The purpose, linked to requirements definition is to organize, scope and define business/operational concepts and rules.
- Logical: this model defines how the information should be organized from a logical point of view, in between the business/operational view and the technical view. The purpose is to develop both feasible and relevant map of rules and data structures consistent with the conceptual model, regardless of the deployment, the data base management system or any other physical consideration (including software assets). C2SIM provides logical model information in an OWL ontology.

- Physical: This model describes the actual implementation for the structure of the data, considering the constraints of the deployment and the real system and technologies. C2SIM defines these physical models in the XML Schema World Wide Web Consortium (W3C) standard.

One of the possible illustrations of the logical models would be a table with data modelling objects such as entities, attributes and relationships as rows and data model levels as columns. The reader should consider this table as a very basic illustration. For example, this table doesn't illustrate that the "entity names" vary from one level to the other one: for the conceptual level "entity names" are non-technical names understandable by end-users, for the logical level they are business/operational names belonging to a dictionary and compliant with a given nomenclature if any, for the physical level they are compliant with coding and naming rules.

Table 1: Data Model Attributes

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	
Foreign Keys		✓	✓
Table Names		✓	✓
Column Name			✓
Column Data Types			✓

In this document, only the logical and physical levels are considered with associated semantic description.

- The logical model is "point of view neutral." The developer is a functional expert, able to understand both the modeling and simulation business/operational requirements and the constraints associated to OWL and the C2SIM transformation XSD. This logical model is expressed in the formalism used for ontologies, i.e. OWL files.
- The physical model is used by technicians and software developers. It is associated with a model expressed in XSD format i.e. a model that can be processed easily to generate classes and attributes and benefit marshalling and un-marshalling mechanisms with almost any kind of software development technology.

1.6 C2SIM Messaging

In order for C2SIM implementations to be interoperable, they must employ consistent message semantics and message syntax. For message semantics, C2SIM has an adapted version of the FIPA specification (see Section 8 below). C2SIM message syntax shall conform to the XML schema resulting from transformation of the C2SIM Ontology (Core or extended) using the procedures specified in Annex: B.

2 References

2.1 SISO Documents

The following SISO documents were used in generating the policies and procedures defined herein. When the following documents are superseded by an approved revision and that causes a conflict with this document, the revision of the below-referenced documents shall supersede this document. These documents are available by through the SISO web site at https://www.sisostds.org/.

Ref:	Document Number	Title	Date
[1]	SISO-PN-010-2014 Revision 1.1	C2SIM Product Nomination	14 Sep 2017
[2]	SISO-STD-011-2014 Version 1.0	Standard for Coalition Battle Management Language	14 Apr 2014
[3]	SISO-STD-007-2008	Standard for Military Scenario Definition Language	Reaffirmed 11 May 2015
[4]	SISO-REF-056-2018	SISO Reference for XML Style Guide	1 August 2018
[5]	SISO-REF-010-2019	Reference for Enumerations for Simulation Interoperability	14 October 2019
[6]	IEEE 1730-2010	Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)	24 January 2011
[7]	SISO-GUIDE-006-2018	Guideline on Scenario Development for Simulation Environments	10 May 2018
[8]	IEEE 1516.2-2010	High Level Architecture – Object Model Template	18 August 2010
[9]	SISO-STD-001-2015	Standard for Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0	10 August 2015

2.2 Other Documents

Ref:	Document Number	Title	Date
[10]	NATO APP-6	MILITARY SYMBOLS FOR LAND BASED SYSTEMS	May 2011
[11]	OWL	OWL Web Ontology Language Reference, W3C	10 Feb 2004
[12]	XML	Extensible Markup Language (XML) 1.0	2008
[13]	FIPA	Foundation for Intelligent Physical Agents, http://www.fipa.org	2005
[14]	Webster	Webster's New World College Dictionary	2018
[15]	MIP JC3IEDM	JC3IEDM, Annexes, and .xsd Domain Values; www.mip-interop.org	2020
[16]	ANSI/X3/SPARC Study Group on Data Base Management Systems: (1975), Interim Report. FDT, ACM SIGMOD bulletin. Volume 7, No. 2	ANSI SPARC Architecture	1975

3 Definitions, Acronyms, and Abbreviations

English words are used in accordance with their definitions in the latest edition of Webster's New Collegiate Dictionary [14] except where special SISO product-related technical terms are required.

3.1 Definitions

Term	Definition
Core	A set of formally defined information elements and associated definitions that are commonly used as part of C2SIM Data, applicable to all or most domains (military, medical, etc.) and/or communities (international, national, services, functional areas, etc.)
Domain Extension	A formally defined information element and associated definition that is commonly used only within a specific domain (military, medical, etc.) and/or community (international, national, services, functional areas, etc.)
Attribute	A characteristic of an object or entity.
Coalition	The preferred collective noun for a C2SIM system-of-systems is "Coalition."
Complex Element	A unit of information (object) for which the definition, identification, representation, and permissible values are specified by means of a set of existing Simple Elements and/or Complex Elements
Compliant	Agreeing with a set of rules, standards, or requirements.
Conceptual Data Model	A conceptual data model is a high-level description of a business's informational needs. It typically includes only the main concepts and the main relationships among them. Typically, this is a first-cut model, with insufficient detail to build an actual database. This level describes the structure of the whole database for a group of users. The conceptual data model, also known as the data model, can be used to describe the conceptual schema when a database system is implemented. It hides the internal details of physical storage and targets on describing entities, datatypes, relationships, and constraints.
Ground Truth	A scenario describes a battlespace containing elements[1] that interact with each other. In support of these interactions, each element may have certain knowledge of the other elements in the scenario. The actual description of the elements is referred to as the ground truth aspect of the item being described because what it describes is the truth in the context of the scenario. These descriptions are exact and not the result of interpretation by the scenario elements.
Information Element or Simple Element	An atomic and inseparable unit of information (object) for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. (ref Organisation for Economic Co-operation)

[1] The term "element" in this section describes battlespace and scenario things and does not equate to the XML element definition as in the rest of the specification.

Term	Definition
Intelligence	1. The product resulting from the collection, processing, integration, analysis, evaluation, and interpretation of available information. 2. Information and knowledge obtained through observation, investigation, analysis, or understanding. (Dictionary of Military and Associated Terms, Joint Publication 1-02, Department of Defense, USA)[2]
Logical Data Model	A logical data model or logical schema is a data model of a specific problem domain expressed independently of a particular database management product or storage technology (physical data model) but in terms of data structures such as relational tables and columns, object oriented classes, or XML tags. This is as opposed to a conceptual data model, which describes the semantics of organization without reference to technology. https://en.wikipedia.org/wiki/Logical_data_model. (update 11/13/2016)
Object	Anything that is perceivable or conceivable.
Optional	Available as a choice but not required.
Order	Document that provides tasking from a source that has authority to impose a requirement.
Order of Battle (OOB)	The hierarchical organization, command structure, strength, disposition of personnel, equipment and formations of an armed force.
Perception	Each scenario element's knowledge of the battlespace. Multiple perception descriptions of a single element will exist in the context of the scenario since multiple elements may have specific knowledge of other elements. These descriptions represent the result of perceptions gathering process and stored within the initialization data set.
Physical Data Model	A physical data model is a representation of a data design as implemented, or intended to be implemented, in a database management system. In the lifecycle of a project it typically derives from a logical data model, though it may be reverse engineered from a given database implementation. A complete physical data model will include all the database artifacts required to create relationships between tables or to achieve performance goals such as indexes, constraint definitions, linking tables, partitioned tables, or clusters. Analysts can usually use a physical data model to calculate storage estimates; it may include specific storage allocation details for a given database system.
Report	Document that provides situational awareness information.
Request	Document that provides tasking from a source that is authorized to ask for task performance but not to require it.

[2] The definitions for intelligence have been modified to allow information and knowledge concerning friendly forces and the environment. The kind of information and knowledge is unspecified in the definition and as such could include COA-relevant data.

3.2 Acronyms and Abbreviations

Acronym/Abbr	Definition
ACL	Agent Communication Language
C-BML	Coalition Battle Management Language
C2	Command and Control
C2SIM	Command and Control to Simulation Interoperability
COA	Course of Action
DIS	Distributed Interactive Simulation
DSEEP	Distributed Simulation Engineering and Execution Process
EXCOM	Executive Committee
FIPA	Foundation for Intelligent Physical Agents
GCC	Geocentric Coordinate
GDC	Geodetic Coordinate
GSD	Guideline on Scenario Development
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers
IFF	Identification, Friend of Foe
JC3IEDM	Joint Consultation, Command, and Control Information Exchange Data Model
LDM	Logical Data Model
LOX	Land Operations Extension
MIP	Multilateral Interoperability Programme
MSDL	Military Scenario Definition Language
OE	Operational Environment
OOB	Order of Battle
OMT	Object Model Template
OWL	Web Ontology Language
PDG	Product Development Group
PDM	Physical Data Model
PDU	Protocol Data Unit
PSG	Product Support Group
RAS	Robotic and Autonomous System
RDF	Resource Description Framework
RPR FOM	Real-time Platform Reference Federation Object Model
SAC	Standards Activity Committee
SISO	Simulation Interoperability Standards Organization

Acronym/Abbr	Definition
SMX	Standard Military Extension
UUID	Universally Unique Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformation

4 Compliance Strategy for the C2SIM Standards Set

The overall compliance strategy of the C2SIM standard is based on adherence to a set of rules and requirements relating to:

1. Creation of extensions to the C2SIM Core ontology and standardized domain extensions (e.g., SMX and LOX)
2. Generation of the XML schema document from the Core ontology and any set of domain extensions
3. Implementation of the XML schema generated from the Core ontology and any set of domain extensions

4.1 Compliance: Rules and Requirements for Creating Domain Extensions to the C2SIM Core Ontology and its Standardized Extensions

The C2SIM Logical Data Model (LDM) shall be described as an ontology specification represented in the Web Ontology Language (OWL).

To be a compliant extension to the C2SIM standard, an extension to the C2SIM Core ontology, plus any standardized extensions (e.g., SMX, LOX) to the Core, shall adhere to the rules specified herein.

C2SIM LDM ontology classes shall not be duplicative. There shall be only one class per concept and only one instance of a given class name within the Core Logical Data Model and all extensions.

C2SIM LDM ontology properties shall not be duplicative. There shall be one definition per property and only one instance of a given property name within each class definition.

A C2SIM-compliant LDM shall use only those OWL constructs for which there are defined transformation rules specified in the C2SIM Standard. Transformation rules are identified in Annex: B. No other OWL constructs are permitted in this version of the standard.

4.2 Compliance: Rules and Requirements for Generating the XML Schema Document from the C2SIM Core Ontology and Any Domain Extensions

The C2SIM Physical Data Model (PDM) shall be described using the XML Schema language. C2SIM PDM implementers work with the XML schema document generated from the C2SIM Core Ontology plus any domain extensions deemed necessary to meet the requirements of a particular C2SIM coalition of systems.

A C2SIM-compliant PDM shall be described by deriving an XML Schema document from the C2SIM Ontology, including any extensions, using the procedures defined in Annex: B.

4.3 Compliance: Implementation of the XML Schema Document Generated from the C2SIM Core and Any Domain Extensions

A C2SIM implementation in a particular C2SIM coalition of systems shall be considered fully compliant with the C2SIM Standard for a set of received or sent C2SIM messages if it implements (i.e., has capability to read for received or write for sent messages) all XML data structures, mandatory and optional, relative to these C2SIM messages in the C2SIM PDM generated for the particular C2SIM coalition of systems.

A C2SIM implementation in a particular C2SIM coalition of systems shall be considered minimally compliant with the C2SIM Standard for a set of received or sent C2SIM messages if it fully implements (i.e., has capability to read for received or write for sent messages) all mandatory XML data structures relative to these messages in the C2SIM PDM generated for the particular C2SIM coalition of systems.

A C2SIM-compliant implementation in a particular C2SIM coalition must not reject any XML document that validates (in the XML sense) against the XML schema document generated for the particular C2SIM coalition of systems.

A “C2SIM message” shall be an XML document conformant with the generated C2SIM PDM.

A “C2SIM message” shall contain one and only one root element ‘Message’.

Only message header data and information shall be used (by an infrastructure) to transport or distribute the messages.

Message processing shall conform to procedures described in Section 8 of this Standard.

5 Core LDM Ontology

Using the OWL ontology as a format for specifying the C2SIM LDM, extensions will be assembled using available tools (e.g. Protégé) to assemble a multi-layered ontology with Core as the lowest layer. Additional data classes and properties of classes are added in the higher layers, creating a composite ontology.

The C2SIM Core Ontology and SMX ontology (that imports the Core ontology) is the normative C2SIM LDM standard. As C2SIM ontologies evolve under control of the C2SIM PSG, an informative text document containing a list of classes and descriptions extracted automatically from each ontology will be posted in the SISO digital library along with the ontology.

An important goal of the C2SIM architecture is to include in the Core LDM only those classes and properties of classes that are expected to be used in all, or nearly all, C2SIM extensions and in the SMX additional classes/subclasses and properties that are expected to be used in all, or nearly all, military extensions. This lets the developers of extensions customize the Core and SMX by further specifying them.

The Core is built around Entity and Action as subcategories because most of the simulations used with C2 systems are agent model-based simulations. Agent-based models allow simulating complex systems and organizations by simulating the actions, interactions and effects of autonomous agents individually and getting as a result, states of the entire whole system that would be impossible to get with a holistic approach. Four basic concepts underlie agent-based models: entity, perception, action and the environment. This could include objects, the state of other entities or the medium (terrain, weather…). An entity gets a perception of its environment, process it according it goals and motivation, and take actions as results of this process, which modifies other entities or objects.

The C2SIM PSG will keep the latest version of the ontology up to date with related standards such as NATO APP-6 and US MIL-STD-2525. The versions as of publication of this standard are APP-6D and MIL-STD-2525D.

Figure 2 through Figure 6 show the top-level structure of the Core and SMX ontologies at the time this standard was presented for initial balloting. The significance of bold-face type in these figures is that it represents a high-level class that was not present in the previous ontology layer. Thus, for example, in the SMX ontology the AbstractOrganization class is inherited without change from the Core but the ForceSide class is added in SMX.

Figure 2: LDM Core Ontology Hierarchy

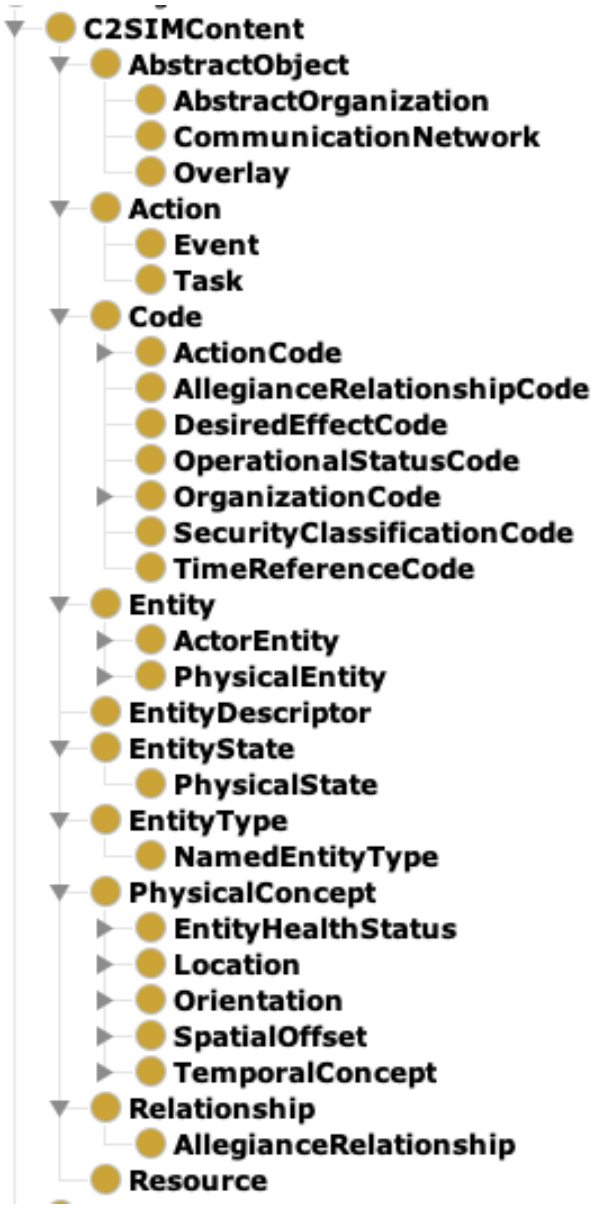

Figure 3: C2SIM Content Hierarchy

Figure 4: InitializationConcept Hierarchy

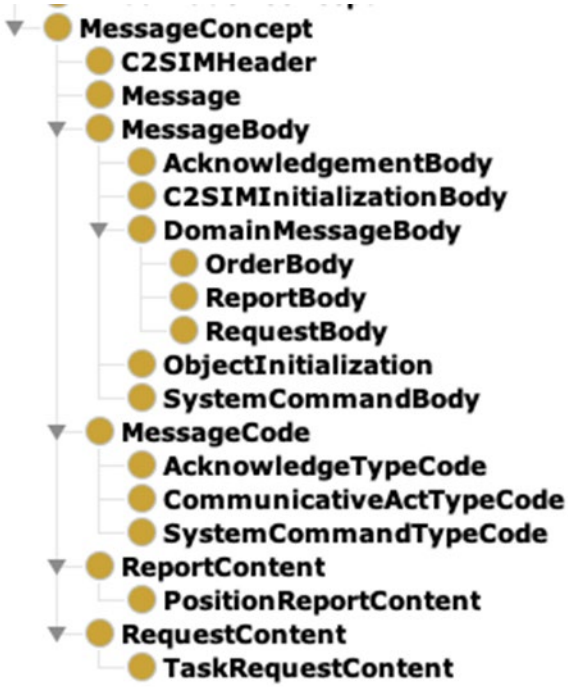

Figure 5: MessageConcept Hierarchy

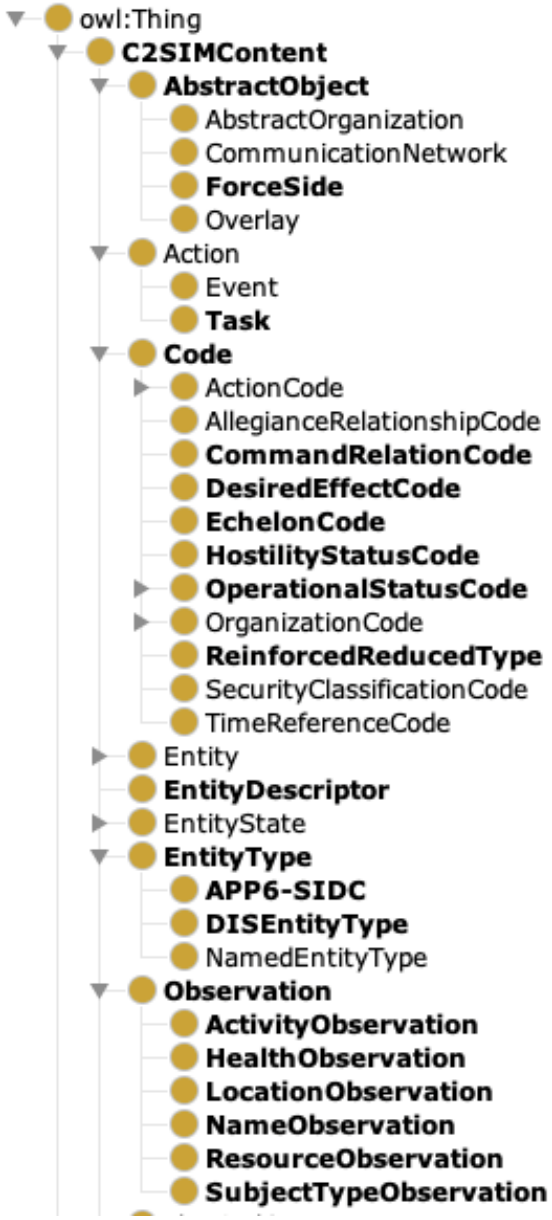

Figure 6: Standard Military Extension hierarchy

6 Extending the C2SIM Ontology

Definition of an extension is expected to be undertaken by a collaborative group that wants to add a C2-simulation interoperation capability for a domain that does not have one. Such a group needs to have strong expertise in the domain of interest and to follow the procedures in this section. These procedures are intended to result in a family of compatible extensions that can be combined as needed to deal simultaneously with any required combination of domains. The standard supports the combination and the compatibility at the following levels:

- At the LDM level, the combination and the compatibility of extensions are supported by the standard with ontologies (see Note below).
- At the PDM level (XLM Schema), only the combination is supported by the standard, by the transformation of the combined ontology (see Note below).

Note that the Physical Data Model (XSD) generated from an ontology composed of (Core + SMX + other Extension) is not necessarily compatible with the Physical Data Model (SMX.XSD) generated from an ontology composed of (Core + SMX). Compatibility means here that any XML document compliant with Core XSD shall be also compliant with the extension XSD. Such compatibility issues shall be addressed at the system level, or system of systems level, when defining a new extension.

Defining the extension is one aspect of the SISO DSEEP process that provides a comprehensive approach to this problem.

While the C2SIM LDM captures core data elements, it is expected that C2SIM implementations will need elements that are not available in the LDM. This standardized methodology will enable C2SIM implementations to interoperate after extensions. The extension methodology is designed to preserve continuity and consistency with C-BML and MSDL while allowing implementations to create new classes and attributes in the data model. C2SIM extensions are required to consider elements defined in the Core and SMX ontologies before creating duplicate classes.

7 Initialization Procedures

This section covers how initialization data is identified and how a sequence of messages initializes the system of systems.

7.1 Initialization Overview

The initialization procedures provided here presume a networked system of systems with at least one C2 system, one simulation system, and generally includes one server that accepts information from any of the systems and distributes it to the other systems on a subscription basis. There may also be a need for marshalling initialization data; this function might be performed by the C2 message server or by a separate marshalling server. This need arises in the case where, before simulation begins, the marshalling system accepts initialization messages from participating systems and aggregates them for distribution to entire system of systems. Operation without marshalling is possible; in this case the aggregated initialization document must be assembled and distributed to participating systems somehow.

7.2 Basic Initialization Data

C2SIM presumes a system of systems: some combination of C2 systems, simulation systems, unmanned systems, and generally includes one or more servers and one master controller that is capable of issuing initialization commands to the network. In basic initialization, the master controller issues a C2SIM Initialization message to the systems as described in Figure 7. C2SIM also supports the concept of a marshalling system or server to which all systems can send partial scenario initialization data, and which combines this into the full scenario description for broadcast in a C2SIM Initialization message.

C2SIM Initialization messages shall contain definitions of individual objects in the C2SIM scenario. The C2SIM LDM ontology defines subsets, i.e. classes, of individuals by listing common properties of the individuals in that class. For an individual in a C2SIM Initialization message to be a valid member of the class, the message must provide values for the mandatory properties that define the class. For example, every individual that is a valid ActorEntity, i.e. able to send and receive C2 orders and reports, must have values for its Universally Unique Identifier (UUID) and EntityType properties provided. Every individual that is to have a physical embodiment in a simulation must have a Location provided for it. The ontology identifies properties that are required for the definition of the class by imposing a cardinality restriction on these properties—every individual in the class must have at least one or exactly one value satisfying the property.

The ontology also specifies properties of individuals in a class that are optional in the declaration of an individual member of the class (example: hasEntityHealthStatus in PhysicalState class). Depending on the requirements of a specific scenario, if users are required to provide values for those optional properties (for some or all instances of the class), such requirements should be stated in a document relative to this specific scenario. For example, a certain C2SIM Coalition of systems may need to declare in its Coalition agreement document that EntityHealthStatus property is mandatory and must be initialized in all ActorEntity individuals.

Order of Battle (OOB): this links each entity in a scenario to its superior and subordinate entities, and through that information to its sibling entities. It can be assembled based on the superior for each entity (hasSuperior in the EntityDescriptor ontology class). Any entity without a hasSuperior value is assumed to be the root of an organization tree.

7.3 Coalition System-of-Systems Synchronization

The following messages, with formats defined in the C2SIM Core Ontology, shall be used to synchronize the initialization process across the C2SIM coalition. Some of these can only be sent by a designated coalition controller:

a) Submit_Initialization: Sent by a C2SIM controller. When C2SIM systems receive this C2SIM_Message, they should submit Object_Initialization messages for object definitions they are configured to simulate. This message contains no data; it serves as a command and is not required if a preconfigured master initialization file is used.

b) Object_Initialization: Sent by each C2 and simulation to a marshalling server on receipt of a Submit_Inititalization C2SIM_Message. This message is not required if a preconfigured master initialization file is used.

c) Share_Scenario: Set by a C2SIM controller to a marshalling server. When this C2SIM_Message is received, the marshalling server sends to all subscribed systems the C2SIM_Initialization message.

d) C2SIM_Initialization: This C2SIM message aggregates initialization data for all systems, as described in Section 7.1. The source of this information could either be a preconfigured C2SIM_Initialization message, or a composite, produced by the marshalling server, of the initialization data in all received Object_Initialization messages.

e) Initialization_Complete: This C2SIM_Message is sent by each participating C2 and simulation system in response the C2SIM_Initialization message, after that system has completed its initialization processing.

f) Start_Scenario: This C2SIM_Message is sent to all subscribed systems indicating that the simulation scenario is to begin.

g) Query_Unit/Query_Init: Sent by a later joiner to get current status (Query_Unit) or initial status (Query_Init) – this C2SIM_Message is optional for server implementation.

h) Pause_Scenario: This C2SIM_Message is sent to all subscribed systems indicating that the simulation should stop such that it can be restarted with a Start_Scenario message.

i) Stop_Scenario: This C2SIM_Message is sent to all subscribed systems indicating that the simulation should be stopped and won't be restarted. All initialization data should be discarded, and all systems should prepare to reinitialize.

Figure 7 shows time synchronization of the various messages as described above.

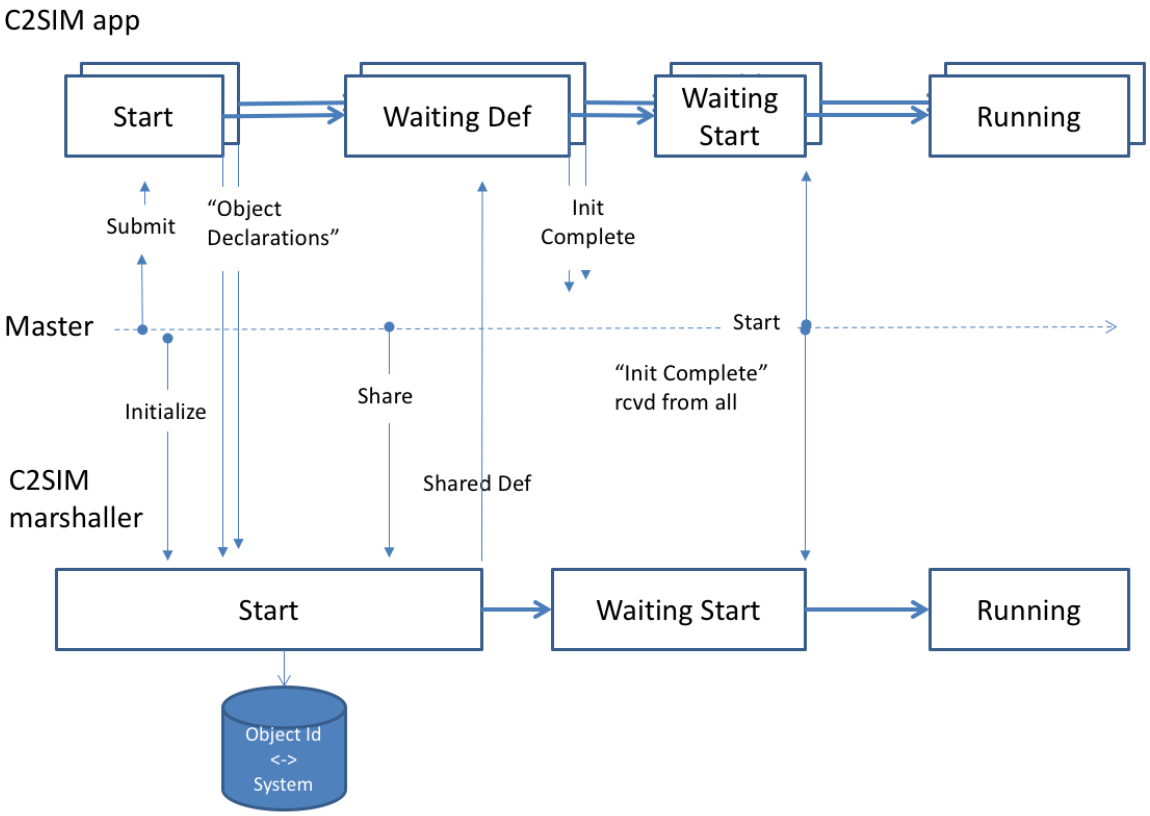

Figure 7: Time Phasing of C2SIM Initialization Messages

8 FIPA-Derived Procedures

8.1 Message Exchange

All Information for C2SIM operations shall be exchanged via messages as defined in the C2SIM Ontology (see Section 5 Core LDM Ontology) using a communication concept conforming to FIPA Agent Communication Language (ACL) Message Structure Specification as shown in Table 2. Each Message contains a header, which contains information required by the FIPA ACL Message Structure Specification. FIPA-compliant messages provide for appropriate delivery and, where required, acknowledgement of a C2SIM Message by means of an outer "envelope" FIPA-compliant message. The DomainMessage is sent between/among “actors,” i.e. software functions such as command and control, simulated actors, and servers. FIPA messages are sent between/among “interfaces,” i.e. software that accepts a network message and passes it to an actor. Initialization and SystemCommand messages are used for startup and synchronization (see Section 7). Each type of MessageBody defined in the ontology shall be represented by an XML document.

The FIPA communication process in Table 2 specifies the behavior of C2SIM communication infrastructure, e.g. passing messages between interfaces. Actors supported by C2SIM should follow good communication procedures as described in the C2SIM Guidance Document, however, specifying their behavior is beyond the scope of the C2SIM Standard.

Table 2: C2SIM Messaging Communicative Acts and Possible Responses

Message Purpose	FIPA Communicative Act according to FIPA Communicative Act Library Specification	Default Interface Response	AcknowledgeTypeCode
Order	Propose	Acknowledge Order	
Acknowledge Order	Accept Proposal	None	ACKRCVD
Refuse Order	Refuse	Orders presented as well-formed messages shall not be refused by interfaces; however, actor behavior may be to reject or ignore them	ACKNOTRECGNZ
Request	Request	Acknowledge Request	
Acknowledge Request	Agree	None	ACKRCVD
Refuse Request	Refuse	Requests presented as well-formed messages shall not be refused by interfaces; however, actor behavior may be to reject or ignore them	ACKNOTRECGNZ
Report	Inform	None	
Initialization	Request	See Figure 7	
System Command	Request	See Figure 7	

Communicative act in the C2SIMHeader shall be one of the values from Table 3.

Table 3: Communicative Acts

Communicative Act
Accept Proposal
Agree
Confirm
Inform
Propose
Refuse
Request

8.2 Protocol and Version

Shall be “SISO-STD-C2SIM-1.0.0” for the initial version. Version Number starts after “C2SIM-“ and is contained in the Core ontology. It shall change upward with ontology updates (see Section 9 below).

9 Maintenance of C2SIM Ontologies

In order to avoid frequent balloting of new standard versions, the C2SIM Ontology and associated Extension Ontologies shall be maintained by the C2SIM PSG using established SISO voting procedures. Ontology revisions shall occur only at in-person or teleconference meetings as established under PSG Voting Procedures defined in the SISO Balloted Products Development and Support Process. The Product Support Group shall assign each revision of each ontology a unique version number.

ANNEX: A BIBLIOGRAPHY (INFORMATIVE)

	Document Number	Title
1	SISO-ADM-002-2008	SISO Policies and Procedures (P&P)
2	SISO-ADM-003-2008	SISO Balloted Products Development Process (BPDP)
3	SISO-ADM-005-2004	Policy for: The Style and Format of SISO documents
4	SISO-PDG-PN-MSDL-2005-002-15	MSDL Product Nomination
5	UN/CEFACT XML Naming and Design Rules	United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) XML Naming and Design Rules Version 2.0 of 17 February 2006
7	NIMA TM 8358.1, 20 September 1990	Datums, Ellipsoids, Grids, and Grid Reference Systems. Edition 1. US National Imagery and Mapping Agency (NIMA)
8	USA NIEM https://www.niem.gov	USA National Information Exchange Model

ANNEX: B C2SIM ONTOLOGY TO XSD (O2S) PROCEDURE (NORMATIVE)

This annex specifies rules for generating an XML schema document from a C2SIM ontology (i.e., the Core ontology plus any extensions, saved as a merged ontology).

B.1 Transformation Restrictions and Rules

B.1.1 Restrictions on Structures Allowed in the Generated XML Schema

Overall, the transformation process must produce an XML schema document that follows the XML design principles identified in the SISO XML Guide [4]. Furthermore, to achieve simplicity and usability in the generated XML schema document, the following restrictions must be observed in the transformation process:

- Transformation Restriction (1): There shall be no declaration of “abstract” types in the generated XML schema document.
- Transformation Restriction (2): There shall be no creation of types through the XML Schema “extension” statement in the generated XML schema document.
- Transformation Restriction (3): Element and type names in the generated XML schema document shall match names provided in the source ontology, with the exception of naming conventions used in the ontology for data and object properties (e.g., the “hasName” property in the ontology is transformed to an element called “Name”).
- Transformation Restriction (4): All elements, complex types, and simple types shall be declared at the top level of the generated XML schema document (i.e., globally defined, no nested anonymous complex types; this style is sometimes referred to as the “flat catalog” or “salami slice” design).[3]
- Transformation Restriction (5): The transformation design or implementation shall not require any change in the C2SIM ontology (Core+extension(s)) with one exception: *annotation properties* may be added or the content of annotation properties may be modified to support the transformation process.
- Transformation Restriction (6): The generated XML schema shall not enable creation of C2SIM messages in XML with “incomplete” data. For example, if a location is expected in the message, the location must be a complete XML structure with respect to the location definition; it should be either a GeocentricCoordinateValue, or a RelativeLocation, or an EntityDefinedLocation, but not an element just having the properties of parent class ‘LocationConcept’.

To simplify the transformation process, some additional restrictions are placed on the target schema constructs:

To simplify the transformation process, some additional restrictions are placed on the target schema constructs:

- Schema Restriction (1): No schema components shall appear in the generated XML schema that are not represented as concepts or properties in the source C2SIM ontology (Core+extension(s)).
- Schema Restriction (2): No anonymous types (aligns with Transformation Restriction (4)) shall be generated in the XML schema by the transformation process.

[3] Note, however, we do not require global declaration of attributes in the generated XML schema document, so this is a slight variation on the flat catalog design that leans toward what is generally called the “Venetian blind” design where *selected* elements and types are globally defined for targeted reuse. On the other hand, the initial transformation logic specified in this annex does not generate any XML attribute declarations in the generated XML schema document.

- Schema Restriction (3): The transformation process shall not produce the “all” compositor in the generated XML schema.

B.1.2 Transformation Rules: Ontology Patterns and Resulting XML Schema Structures

This section identifies a number of “rules” that the transformation process must satisfy in generating an XML schema structure for practical application of the C2SIM standard:

- If a class has subclasses, the schema component generated from that class shall have respective schema components for each of these subclasses.
- If a class has subclasses, the transformation process shall generate an XML model group corresponding to the defining properties of that parent class (for use in generating the “inherited” information in the XML constructs corresponding to each subclass).
- If a class has subclasses, the XML construct corresponding to that parent class shall use the XML “choice” compositor to allow selection of the XML structure corresponding to any direct subclass of that parent class.
- The transformation process shall generate an XML simple type from each data property in the source ontology.
- The transformation process shall generate an XML complex type from each object property in the source ontology.
- The transformation shall generate XML minOccurs and maxOccurs cardinality values in the XML schema according to the following rules:
    - *minQualifiedCardinality* set to some value N in the ontology becomes minOccurs=”N” in the XML schema
    - *maxQualifiedCardinality* set to some value N in the ontology becomes maxOccurs=”N” in the XML schema
    - *qualifiedCardinality* set to some value N in the ontology becomes minOccurs=”N” and maxOccurs=”N” in the schema (note, when N=1, the transformation process can choose to leave off the minOccurs and maxOccurs attributes since the XML schema default value for these is “1”)
    - existential restriction *someValuesFrom* in the ontology becomes minOccurs=”1” and maxOccurs=”unbounded” in the XML schema
    - universal restriction *allValuesFrom* in the ontology becomes minOccurs=”0” and maxOccurs=”unbounded” in the XML schema
- The C2SIM ontology design follows the general convention of naming properties with prefixes such as “has” or “is” for readability. However, this convention is seldom used in XML schema documents In the naming of simple types and complex types in the generated XML schema document, the transformation process shall strip prefixes such as “has” or “is” from the names of data properties and object properties in the source ontology.
- The transformation process shall append the string “Type” to the end of the names of simple types and complex types in the generated XML schema document.
- For every simple type and complex type generated in the XML schema document, the transformation process shall generate a corresponding element declaration of that type. Generated XML elements shall be used by reference (e.g., using the attribute ‘ref=”elementName”’) in other XML schema constructs in the generated schema document.

- The transformation process shall produce:
  <xs:annotation> <xs:documentation> </xs:documentation> </xs:annotation>
  blocks in the generated XML schema document from *rdfs:comment* statements in the source ontology.
- The transformation process shall produce:
  <xs:annotation> <xs:documentation>…</xs:documentation> </xs:annotation>
  blocks in the generated XML schema document, with the content ("…" above) giving the ontology namespace from which this component in the generated XML schema was derived.
- The C2SIM ontology declares named individuals as members of "Code" classes. The transformation process shall generate a simple type, named for the name of the class but with "Type" appended to the name of the class, declared as an enumeration having string values corresponding to the names of the individuals belonging to a "Code" class in the source ontology.
- The C2SIM ontology uses numeric range restrictions in the declaration of certain data properties. The transformation process shall generate simple types corresponding to these data properties, applying the same range restrictions in the declaration of the type in the generated XML schema document.
- The C2SIM ontology uses string patterns in the declaration of certain string-valued data properties. The transformation process shall generate simple types corresponding to these data properties, applying the same string pattern in the declaration of the type in the generated XML schema document.
- The C2SIM ontology may have ranges defined on data properties. If no range declaration is found on the given property, the transformation logic shall follow the chain of parent properties until a range restriction is found in the ontology, and then generate a simpleType in the XML schema document corresponding to the subject data property as a restriction on the datatype represented by the range found on one of its ancestor properties. If no range is found, the transformation logic shall create the XML schema simpleType declaration as a restriction on xs:string for the subject data property found in the source ontology.
- All complex elements (for example, sequence and choice) in the schema resulting from the transformation process shall have their internal groups and elements ordered in ascending alphanumeric order of tags, with groups first followed by elements.

B.2 Examples of Application of the Transformation Rules to OWL Ontology Patterns

This section provides examples of application of the transformation process as specified above to various "patterns" of declarations found in the Resource Description Framework (RDF)/XML representation of the C2SIM ontology. This format is generated by various tools, such as the free Protégé ontology editing tool, and provides a useful organization of the various declarations in the ontology into datatypes, object properties, data properties, classes, and individuals. Since this representation of the ontology is in an XML format, an exemplar transformation was created in the Extensible Stylesheet Language Transformation (XSLT) XML language. The example transformation processes the declarations from the ontology file in the following order: datatypes, data properties, object properties, and classes (including classes having named individuals). The order of declared XML schema components in the generated XML schema file corresponds to the processing order in the XSLT transformation logic.

The following discussion shows the target XML schema structures that need to be generated by the transformation process. According to the specified rules, the following transformations are applied to the various declarations found in the ontology:

- datatypes and data properties declared in the ontology are transformed into XML schema *simpleType* declarations
- object properties declared in the ontology are transformed into XML schema *element* declarations

- class declarations in the ontology are transformed into combinations of XML schema *named model group* declarations, *complexType* declarations (with “choice” compositor for parent classes), and *element* declarations, and
- named individuals of a particular class declared in the ontology are transformed into an XML schema *simpleType* declaration using the class name (with “Type” appended to the name) as the name of the *simpleType* and the identifiers of the named individuals as enumerated values of that *simpleType*.

In these examples, the C2SIM namespace “http:://www.sisostds.org/ontologies/C2SIM” is the default namespace in the generated XML schema examples.

B.2.1 Datatypes

In Protégé, there are a large number of built-in datatypes available for formatting content of the information model. Many of the built-in types come from the XML Schema language specification, such as xs:string, xs:integer, etc. (where “xs” designates the namespace prefix for the XML Schema namespace; namely, “http://www.w3.org/2001/XMLSchema”). The Protégé tool also provides a capability for users to create custom datatypes, often based on restrictions of the built-in types.

String Pattern in the Ontology: for example, consider the following declaration of a IsoDateTime data format defined as a pattern restriction on the xs:string built-in type:

```
<rdfs:Datatype rdf:about="http://www.sisostds.org/ontologies/C2SIM#IsoDateTime">
    <owl:equivalentClass>
      <rdfs:Datatype>
        <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:withRestrictions rdf:parseType="Collection">
          <rdf:Description>
            <xsd:pattern>[0-9]{4}-[0-9]{2}-[0-9]{2}[T]{1}[0-9]{2}:[0-9]{2}:[0-9]{2}:[Z]{1}</xsd:pattern>
          </rdf:Description>
        </owl:withRestrictions>
      </rdfs:Datatype>
    </owl:equivalentClass>
  </rdfs:Datatype>
```

The example transformation logic creates a simpleType declaration in the generated XML schema as follows:

```
<xs:simpleType name="IsoDateTimeType">
  <xs:annotation>
    <xs:documentation>http://www.sisostds.org/ontologies/C2SIM#IsoDateTime</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{4}-[0-9]{2}-[0-9]{2}[T]{1}[0-9]{2}:[0-9]{2}:[0-9]{2}:[Z]{1}"/>
  </xs:restriction>
</xs:simpleType>
```

Numeric, with Range Restrictions: as another example, consider the following definition of a custom datatype for a latitude numeric value:

```
<rdfs:Datatype rdf:about="http://www.sisostds.org/ontologies/C2SIM#latitude">
  <owl:equivalentClass>
    <rdfs:Datatype>
      <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
      <owl:withRestrictions rdf:parseType="Collection">
        <rdf:Description>
          <xsd:minInclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#double">-90.0</xsd:minInclusive>
        </rdf:Description>
        <rdf:Description>
          <xsd:maxInclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#double">90.0</xsd:maxInclusive>
        </rdf:Description>
      </owl:withRestrictions>
    </rdfs:Datatype>
  </owl:equivalentClass>
</rdfs:Datatype>
```

The transformation logic creates a *simpleType* declaration in the generated XML schema as follows:

```
<xs:simpleType name="latitudeType">
   <xs:annotation>
      <xs:documentation>http://www.sisostds.org/ontologies/C2SIM#latitude</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:double">
      <xs:minInclusive value="-90.0"/>
      <xs:maxInclusive value="90.0"/>
   </xs:restriction>
</xs:simpleType>
```

In all cases, the string “Type” is appended to the name of the datatype declared in the ontology to form the name of the corresponding simpleType in the generated XML schema.

B.2.2 Data Properties

A data property (referred to in OWL as a DatatypeProperty) appears in the ontology as follows:

```
<owl:DatatypeProperty rdf:about="http://www.sisostds.org/ontologies/C2SIM#hasDepth">
   <rdfs:subPropertyOf rdf:resource="http://www.sisostds.org/ontologies/C2SIM#hasDistance"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
   <rdfs:comment>This is a measure of distance below mean sea level, i.e. positive is down.</rdfs:comment>
</owl:DatatypeProperty>
```

The example transformation logic creates a *simpleType* declaration and a corresponding *element* declaration in the generated XML schema as follows:

```
<xs:simpleType name="DepthType">
    <xs:annotation>
        <xs:documentation>This is a measure of distance below mean sea level, i.e. positive is down.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:double"/>
</xs:simpleType>
<xs:element name="Depth" type="c2s:DepthType"/>
```

Note that the example transformation logic creates an xs:annotation/xs:documentation element structure in the generated XML schema from the “rdfs:comment” declaration in the ontology (this is true whenever an “rdfs:comment” declaration appears anywhere in the ontology). The C2SIM ontology design generally follows the convention of naming properties with prefixes such as “has” or “is” for readability. However, this convention is seldom used in XML schema documents, so the example XSLT logic strips off such prefixes. Unfortunately, this means that the XSLT logic needs to have a list of such prefixes hard-coded into the logic so it can identify and remove such prefixes.

In the above example, the *hasDepth* data property has a declared range of xs:double (or, more precisely, “http://www.w3.org/2001/XMLSchema#double”), so the transformation logic uses that datatype directly in the generated XML schema simpleType declaration. If no range declaration is found on the given property, the example XSLT transformation logic follows the chain of parent properties until a range restriction is found in the ontology, and then generates the simpleType in the XML schema document as a restriction on the datatype represented by that range. If no range is found, the logic creates the XML schema simpleType declaration as a restriction on xs:string.

B.2.3 Object Properties

An object property appears in the ontology as follows:

```
<owl:ObjectProperty rdf:about="http://www.sisostds.org/ontologies/C2SIM#hasAction">
   <rdfs:range rdf:resource="http://www.sisostds.org/ontologies/C2SIM#Action"/>
   <rdfs:comment>A reference to an Action (an Event or Task).</rdfs:comment>
</owl:ObjectProperty>
```

In the case of an object property, the range refers to a class in the ontology. In the processing of class declarations, the transformation logic creates corresponding XML schema *complexType* declarations as needed (this is discussed in the next subsection). Therefore, no *complexType* declaration needs to be generated here, just the corresponding element of that type. The example transformation logic creates an *element* declaration in the generated XML schema as follows:

```
<xs:element name="Action" type="c2s:ActionType">
    <xs:annotation>
        <xs:documentation>A reference to an Action (an Event or Task).</xs:documentation>
    </xs:annotation>
</xs:element>
```

For the benefit of the transformation logic, it is important to note that the ontology developer shall specify the range on each *object* property declared in the ontology.

B.2.4 Classes

Declarations of named individuals as members of a declared class appear as follows in the ontology:

```
<owl:NamedIndividual rdf:about="http://www.sisostds.org/ontologies/C2SIM#ACK">
   <rdf:type rdf:resource="http://www.sisostds.org/ontologies/C2SIM#AcknowledgeTypeCode"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.sisostds.org/ontologies/C2SIM#RCVD">
   <rdf:type rdf:resource="http://www.sisostds.org/ontologies/C2SIM#AcknowledgeTypeCode"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.sisostds.org/ontologies/C2SIM#READ">
   <rdf:type rdf:resource="http://www.sisostds.org/ontologies/C2SIM#AcknowledgeTypeCode"/>
</owl:NamedIndividual>
```

Here, the ontology declares three named individuals as members of the AcknowledgeTypeCode class. The transformation logic creates an XML schema *simpleType* declaration having each of the identifiers of the individuals as enumeration values as follows:

```
<xs:simpleType name="AcknowledgeTypeCodeType">
    <xs:annotation>
        <xs:documentation>Enumeration of different types of acknowledgements.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:enumeration value="ACK"/>
        <xs:enumeration value="RCVD"/>
        <xs:enumeration value="READ"/>
    </xs:restriction>
</xs:simpleType>
```

A more typical example of a class declaration in the ontology contains a number of property axioms, as shown in the following declaration of the PhysicalEntity class:

```
<owl:Class rdf:about="http://www.sisostds.org/ontologies/C2SIM#PhysicalEntity">
      <rdfs:subClassOf rdf:resource="http://www.sisostds.org/ontologies/C2SIM#Entity"/>
      <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.sisostds.org/ontologies/C2SIM#hasCurrentState"/>
            <owl:qualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:qualifiedCardinality>
            <owl:onClass rdf:resource="http://www.sisostds.org/ontologies/C2SIM#PhysicalState"/>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.sisostds.org/ontologies/C2SIM#hasMarking"/>
            <owl:maxQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:maxQualifiedCardinality>
            <owl:onDataRange rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:comment>An Entity that has physical/kinematic properties.</rdfs:comment>
   </owl:Class>
```

The example transformation logic handles several special cases in the processing of class declarations. The PhysicalEntity class provides an example of the most general case. To examine the transformation process, we begin by showing the generated XML schema constructs and then discuss each of the components individually. For the above declaration of the PhysicalEntity class in the ontology, the example transformation logic produces the following XML schema constructs:

```
<xs:group name="PhysicalEntityGroup">
    <xs:annotation>
        <xs:documentation>An Entity that has physical/kinematic properties.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element ref="c2s:CurrentState" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="c2s:Marking" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="PhysicalEntityType">
    <xs:choice>
        <xs:element ref="c2s:CulturalFeature"/>
        <xs:element ref="c2s:EnvironmentalObject"/>
        <xs:element ref="c2s:GeographicFeature"/>
        <xs:element ref="c2s:MapGraphic"/>
        <xs:element ref="c2s:Person"/>
        <xs:element ref="c2s:Platform"/>
    </xs:choice>
</xs:complexType>
<xs:element name="PhysicalEntity" type="c2s:PhysicalEntityType"/>
```

First, we see the declaration of a named model group PhysicalEntityGroup containing a sequence of exactly 1 CurrentState element and zero to 1 Marking element. The cardinalities are obtained from the owl:QualifiedCardinality and owl:maxQualifiedCardinality declarations in the definition of the PhysicalEntity class. The example transformation created a named model group because it determined that the PhysicalEntity class has subclasses (e.g., CulturalFeature, EnvironmentalObject, etc.). These subclasses inherit the property axioms of their parent, so the example transformation logic creates this named model group to be used in the XML schema declarations that correspond to the structure of those subclasses. For example, the CulturalFeature class is declared in the ontology as:

```
<owl:Class rdf:about="http://www.sisostds.org/ontologies/C2SIM#CulturalFeature">
    <rdfs:subClassOf rdf:resource="http://www.sisostds.org/ontologies/C2SIM#PhysicalEntity"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://www.sisostds.org/ontologies/C2SIM#hasConsumableMaterial"/>
        <owl:minQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">0</owl:minQualifiedCardinality>
        <owl:onClass rdf:resource="http://www.sisostds.org/ontologies/C2SIM#ConsumableMaterial"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment>Any man-made feature in the environment</rdfs:comment>
  </owl:Class>
```

The example transformation logic creates a XML schema complexType declaration for CulturalFeature as follows:

```
<xs:complexType name="CulturalFeatureType">
    <xs:annotation>
        <xs:documentation>Any man-made feature in the environment</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:group ref="c2s:PhysicalEntityGroup"/>
        <xs:group ref="c2s:EntityGroup"/>
        <xs:element ref="c2s:ConsumableMaterial" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

Here we see reference to the named model group for its superclass (PhysicalEntity) as well as for a higher ancestor in the class hierarchy (Entity). The complexType also has an element corresponding to the CulturalFeature class's own property axiom (declared in the ontology as "hasConsumableMaterial minimum 0 ConsumableMaterial").

Returning to the XML schema constructs generated from the PhysicalEntity class definition, the example transformation logic generates a "choice" compositor in the generation of an XML schema complexType. The elements in the choice compositor correspond to the subclasses having PhysicalEntity as their direct superclass.

Finally, the example transformation logic creates an XML schema element declaration for the PhysicalEntity since that class is not declared as the range of any object property.

This description of an example application of the transformation rules and restrictions addresses RDF/XML patterns currently generated by the Protégé ontology editing tool for the initial version of the C2SIM ontology. Any particular version of the C2SIM ontology (Core+extension(s)) will have a fixed set of patterns appearing in the RDF/XML representation generated by the Protégé ontology editing tool.