



NGA.STND.0064_1.0_NEO
2017-09-15

NGA STANDARDIZATION DOCUMENT

National System for Geospatial Intelligence Enterprise Ontology (NEO) Standard (2017-09-15)

Edition 1.0

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

NATIONAL CENTER FOR GEOSPATIAL INTELLIGENCE STANDARDS

Table of Contents

Introduction	v
1 Scope	6
2 Conformance.....	6
2.1 Conformance Requirements.....	6
2.2 Abstract Test Suite	7
3 References.....	7
3.1 Normative	7
3.2 Informative	7
4 Terms, Definitions, and Acronyms	8
4.1 Terms and Definitions.....	8
4.2 Acronyms.....	12
4.3 Presentation Font	13
5 Ontology Specification.....	13
5.1 Introduction	13
5.2 NEO Information Model	14
5.2.1 Introduction	14
5.2.2 Diagram of the Information Model	15
5.2.3 Ontology.....	16
5.2.4 EntityClass	17
5.2.5 DisjointClasses Axiom.....	18
5.2.6 EntityProperty and its Subclasses.....	18
5.2.7 DocumentationProperty.....	21
5.2.8 DataType.....	21
5.3 NEO Representation using Semantic Web Languages	26
5.3.1 The Semantic Web.....	26
5.3.2 Selecting OWL Constructs for Representation of NEO Content.....	28
5.3.3 Representing NEO Information Model Concepts in OWL.....	28
5.3.4 Unique Identifiers in OWL: IRIs	29
5.3.5 NEO Structural Elements in OWL	29
5.3.6 NEO Documentation Properties in OWL	30
5.3.7 NEO Datatypes in OWL	30
5.4 NEO Content Encodings.....	31
5.4.1 Introduction	31
5.4.2 Namespace and Identifiers.....	31
5.4.3 General NEO Encoding.....	33
5.4.4 General Encoding of Datatypes	42
5.4.5 Technology-specific NEO Encodings	51
6 Governance and Publication	56
6.1 Introduction	56
6.2 Governance	56
6.3 Publication	56
6.3.1 Introduction	56
6.3.2 Publication of NEO Content as a Technical Artifact	56
6.3.3 Publication of NEO Content as REST API-accessible Resources.....	57
Annex A – Conformance (Normative)	63
A.1 Introduction	63
A.1.1 Terms and Definitions.....	63
A.1.2 Conformance Testing Methodology.....	64
A.1.3 Logical Structure of the Abstract Test Suite	65
A.2 Abstract Test Suite for the NSG Enterprise Ontology (NEO)	67
A.2.1 Test Module for Conformance to Ontology Structure	67
A.2.1.1 Test Case for Ontology Dependency(ies)	67

A.2.1.2	Test Module for Components with Identity (IRIs)	67
A.2.1.2.1	Test Case for Ontology with IRI	67
A.2.1.2.2	Test Case for Entity Classes with IRIs	68
A.2.1.2.3	Test Case for Entity Properties with IRIs	68
A.2.1.2.4	Test Case for DisjointClasses with IRIs	68
A.2.1.3	Test Module for Generalization (Subclass) Hierarchy	68
A.2.1.3.1	Test Case for Top Entity Classes	69
A.2.1.3.2	Test Case for Generalization Relationships	69
A.2.1.4	Test Module for Disjointness Axioms	69
A.2.1.4.1	Test Module for Disjoint Collection or List	69
A.2.1.4.1.1	Test Case for DisjointClasses Collection	70
A.2.1.4.1.2	Test Case for DisjointClasses List	70
A.2.1.4.2	Test Case for Members of DisjointClasses	70
A.2.1.4.3	Test Case for Skolemized IRIs	70
A.2.1.5	Test Module for Properties	70
A.2.1.5.1	Test Case for Entity Attributes	71
A.2.1.5.2	Test Case for Entity Relationships	71
A.2.1.5.3	Test Case for Property Domain	71
A.2.1.5.4	Test Case for Property Range	71
A.2.1.5.5	Test Case for Property Inverse	72
A.2.2	Test Module for Documentation of Semantics	72
A.2.2.1	Test Module for Ontology Documentation	72
A.2.2.1.1	Test Case for Ontology Version Information	72
A.2.2.1.2	Test Case for Ontology Label	72
A.2.2.1.3	Test Case for Ontology Name	73
A.2.2.1.4	Test Case for Ontology Alias	73
A.2.2.1.5	Test Case for Ontology Definition Note	73
A.2.2.1.6	Test Case for Ontology Source Reference	73
A.2.2.1.7	Test Case for Ontology Source Title	73
A.2.2.2	Test Module for Ontology Component Documentation	74
A.2.2.2.1	Test Case for Abstract Ontology Component	74
A.2.2.2.2	Test Case for Ontology Component Label	74
A.2.2.2.3	Test Case for Ontology Component Name	74
A.2.2.2.4	Test Case for Ontology Component Alias	74
A.2.2.2.5	Test Case for Ontology Component Definition Note	75
A.2.2.2.6	Test Case for Ontology Component Source Reference	75
A.2.2.2.7	Test Case for Association Name	75
A.2.2.2.8	Test Case for Constraint	75
A.2.2.2.9	Test Case for Ontology Component Part-of	76
A.2.3	Test Module for Datatype Conformance	76
A.2.3.1	Test Module for Primitive Datatypes	76
A.2.3.1.1	Test Case for IRI Datatype	76
A.2.3.1.2	Test Case for Boolean Datatype	76
A.2.3.1.3	Test Case for DateTime Datatypes	77
A.2.3.1.4	Test Case for CharacterString Datatype	77
A.2.3.1.5	Test Case for LocalizedCharacterString Datatype	77
A.2.3.1.6	Test Case for LocalizedContinuousString Datatype	77
A.2.3.1.7	Test Case for IANALanguageSubtag Datatype	77
A.2.3.1.8	Test Case for Real Datatype	78
A.2.3.1.9	Test Case for Decimal Datatype	78
A.2.3.1.10	Test Case for Integer Datatypes	78
A.2.3.2	Test Module for Measure Datatypes	78
A.2.3.2.1	Test Case for Measure Value	78
A.2.3.2.2	Test Case for Measure Unit	79
A.2.3.3	Test Module for Enumerated Types	79
A.2.3.3.1	Test Case for Enumeration	79

A.2.3.3.2	Test Case for Codelist.....	79
A.2.3.3.3	Test Case for Listed Value	79
A.2.3.4	Test Module for Complex Datatypes	80
A.2.3.4.1	Test Case for Complex Datatype	80
A.2.3.4.2	Test Case for Datatype Union	80
A.2.3.4.3	Test Case for Datatype Meta	80
Annex B – ICS Pro Forma (Normative)		81
B.1	Introduction	81
B.2	ICS Pro Forma for the NEO.....	81
Annex C – NEOX Utility Ontology for NSG Enterprise Ontology (Normative).....		84
C.1	Introduction	84
C.2	IRIs.....	84
C.3	Concepts.....	84
C.4	Publication of NEOX	84
Annex D – Inspecting NEO Content (Informative)		85
D.1	Introduction	85
D.2	NEO Content Inspection using the Protégé Ontology Tool	85
D.2.1	Protégé: An Open-Source Ontology Tool for Viewing W3C OWL ontologies	85
D.2.2	Viewing NEO Content using Protégé and its Plug-ins.....	85
Annex E – UML Primer (Informative).....		96
E.1	UML Notations	96
E.2	UML Model Relationships.....	96
E.2.1	Associations.....	96
E.2.2	Navigation	96
E.2.3	Generalization.....	97
E.2.4	Instantiation / Dependency	97
E.2.5	Roles.....	97
E.3	UML Model Stereotypes	98

Table of Figures

Figure 1 – Overview of the NEO Information Model.....	15
Figure 2 – Model of the NEO Datatypes	23
Figure 3 – The Semantic Web Stack.....	27
Figure 4 – Range-based Alternative Encodings for EntityAttribute.....	38
Figure 5 – NEO Datatype Hierarchy (Upper-level).....	42
Figure 6 – OWL2 RDF/XML Encoding: Entity Class Aerodrome and its Disjoint Subclasses	52
Figure 7 – OWL 2 RDF/XML Encoding: Subclasses of Aerodrome	53
Figure 8 – N-Triples Encoding: Entity Class LandAerodrome	55
Figure 9 – N-Triples Encoding: Disjoint Subclasses of Aerodrome.....	55
Figure 10 – Resource Representation for NEO Enumeration ApronAccessibilityStatusType	60
Figure 11 – Resource Representation for NEO Enumeration ApronAccessibilityStatusType_ConceptScheme.....	61
Figure 12 – Resource Representation for NEO Listed Value ApronAccessibilityStatusType/locked	62
Figure 13 – Resource Representation for NEO Listed Value ApronAccessibilityStatusType/lockedOpen.....	62
Figure 14 – Structure of the Abstract Test Suite for the NSG Enterprise Ontology	66
Figure 15 – NEO Described on the Active Ontology Tab of the Protégé Tool.....	86
Figure 16 – Protégé NEO Hierarchy View with Definition of the Class FeatureEntity	88
Figure 17 – OntoGraf Plug-in View of NEO ActorEntity Hierarchy with Relationships	89
Figure 18 – Protégé Class View of NEO Building including its Subclasses, Annotations, and Property Cardinalities ..	90
Figure 19 – Protégé View of Object Properties for Building (with Building.featureFunction Selected).....	91
Figure 20 – NEO Complex Datatype BuildingFeatureFunctionCodeMeta.....	93
Figure 21 – Protégé View of NEO Object Property BuildingFeatureFunctionCodeMeta.values	94
Figure 22 – Protégé Class View of the NEO Enumeration VoidValueReason	95
Figure 23 – UML Notation	96
Figure 24 – UML Roles	97

Table of Tables

Table 1 – Normative References.....	7
Table 2 – Informative References	8
Table 3 – Definitions Applicable to this Standard	8
Table 4 – Definition of Ontology and its Properties	17
Table 5 – Definition of EntityClass and its Properties.....	18
Table 6 – Definition of DisjointClasses.....	18
Table 7 – Definition of EntityProperty and its Properties	19
Table 8 – Definition of EntityAttribute and its Properties	20
Table 9 – Definition of EntityRelationship and its Properties	20
Table 10 – Encoding Elements for the Ontology	35
Table 11 – Encoding Elements for EntityClass	36
Table 12 – Encoding Elements for DisjointClasses.....	37
Table 13 – Encoding Elements for EntityAttribute (with PrimitiveDatatype Range).....	39
Table 14 – Encoding Elements for EntityAttribute (with non-PrimitiveDatatype Range).....	40
Table 15 – Encoding Elements for EntityRelationship.....	41
Table 16 – Encoding Elements for PrimitiveDatatype	44
Table 17 – Encoding Elements for MeasureDatatype	45
Table 18 – Encoding Elements for EnumeratedType.....	46
Table 19 – Encoding Elements for EnumeratedTypeScheme.....	47
Table 20 – Encoding Elements for ListedValue Datatype	48
Table 21 – Encoding Elements for DatatypeUnion.....	49
Table 22 – Encoding Elements for DatatypeMeta	50
Table 23 – Terms and Definitions for Conformance Testing	63
Table 24 – Concept(s) in the NEOX Ontology.....	84

Introduction

The NSG Enterprise Ontology (NEO) Standard document (“NEO Standard”) defines the specification for a logical theory that defines domain concepts used in Geospatial Intelligence (GEOINT) information shared in the U.S. National System for Geospatial Intelligence (NSG). The ontology contains entity classes and properties, including relationships. The ontology is formalized using the representation language defined in the *World Wide Web Consortium (W3C) Web Ontology Language, Second Edition (OWL 2)*. This NEO Standard specifies the ontology information model, two encoding patterns, and a governance process.

The content of the NSG Enterprise Ontology (“NEO content”) is the OWL 2 ontology that specifies the entity classes and properties to be used for the representation of GEOINT information. The NEO content is presented separately from the NEO Standard in officially published technical artifacts. The technical artifacts are implemented in two of the W3C encodings defined for the Semantic Web – Resource Description Framework (RDF) XML and N-Triples. The NEO content is derived from the NSG Application Schema (NAS), which is the logical model for geospatial data in the NSG enterprise.

The NEO enables the semantics (*i.e.*, meaning) of GEOINT data published on the Web to be represented based on well-known International Standards and W3C Recommendations.¹ In this way, the NEO supports collaborative efforts across the U.S. National System for Geospatial Intelligence (NSG) to build a linked store of GEOINT data with integrated, machine-processable semantics. The specific purpose of the NEO logical theory is to enable the representation of GEOINT information (especially data instances) in a common semantic framework, which supports improved management, search, retrieval, and utilization of those data resources. The NEO promotes data interoperability between applications that require a rich description of intelligence information based on standards, in order to enable data integration, categorization, indexing, search, query answering, constraint assertion, logical inference, and/or Web services.

Both the NEO Standard and NEO content are developed and managed under the authority of the National Geospatial-Intelligence Agency (NGA) as a Standards Development Organization (SDO). The NEO Standard and the two NEO content encodings are published as registered technical artifacts in the online NSG-unique Standards Register of the NSG Standards Registry.

Revision History

Description	Date	Edition
Initial Edition	09/15/2017	1.0

¹*Data on the Web Best Practices, a W3C Recommendation* (31 January 2017). Latest version available online at: <http://www.w3.org/TR/dwbp/>. Guidance on the production of data instances falls outside the scope of this standard.

1 Scope

The NSG Enterprise Ontology (NEO) Standard document (“NEO Standard”) defines the specification for a logical theory that defines domain concepts used in Geospatial Intelligence (GEOINT) information shared in the U.S. National System for Geospatial Intelligence (NSG).² The ontology contains entity classes and properties, including relationships. The ontology is formalized using the representation language defined in the *World Wide Web Consortium (W3C) Web Ontology Language, Second Edition (OWL 2)*. The NEO enables the semantics (*i.e.*, meaning) of GEOINT data published on the Web to be represented based on well-known International Standards and W3C Recommendations. This NEO Standard specifies the information model for the ontology, along with two encoding patterns; these are presented in Section 5.

The content of the NSG Enterprise Ontology (“NEO content”) is the OWL 2 ontology that specifies the entity classes and properties to be used for the representation of GEOINT information. The NEO content is presented separately from the NEO Standard in officially published technical artifacts. The technical artifacts are implemented in two of the W3C encodings defined for the Semantic Web – Resource Description Framework (RDF) XML and N-Triples.³

The NEO Standard supports collaborative efforts across the NSG to build a linked store of GEOINT data with machine-processable semantics. This common semantic framework supports improved management, search, retrieval, and utilization of GEOINT. The NEO content promotes data interoperability between applications that enable data integration, categorization, indexing, search, query answering, constraint assertion, logical inference, and/or Web services.

Guidance on the production of data instances falls outside the scope of this standard.

Both the NEO Standard and NEO content are developed and managed under the authority of the National Geospatial-Intelligence Agency (NGA) as a Standards Development Organization (SDO). The NEO Standard and NEO content evolve in response to NSG community requirements. The NEO Standard and the two NEO content encodings are published as registered technical artifacts in the online NSG-unique Standards Register of the NSG Standards Registry hosted by the National Geospatial-Intelligence Agency (NGA). The NSG Standards Registry is the single authoritative source for the NEO Standard and for the technical artifacts encoding the NEO content in RDF/XML and N-Triples format. The governance process is specified in Section 6.

The NGA is the authority for promulgating the NEO Standard and its accompanying technical artifacts encoding the NEO content for use by the U.S. Department of Defense (DoD), U.S. Intelligence Community (IC), and U.S. civil federal agencies.

2 Conformance

2.1 Conformance Requirements

Any product claiming conformance to the NEO (including the NEO Standard and associated NEO content) shall pass all the requirements stated in the abstract test suite (ATS) in Annex A, which enumerates the specific elements of conformance.

This standard defines a single class of conformance: Conformance Class A – Conformance for the Complete NEO.

Conformance may be claimed for data or software products, for services, and by specifications, including functional standards. The kinds of products expected to make use of, and claim conformance to, the NEO Standard and registered NEO content include:

- representations of GEOINT data specified using concepts in NEO content;
- search applications that use machine-processable semantics for indexing and querying (including query expansion and extension);
- data integration services that leverage a reference ontology supporting inference and constraints; and
- automated reasoning tools that provide services such as classification, satisfiability, entailment, consistency testing, conjunctive query answering, and retrieval of data instances.

Products that claim conformance to the NEO shall use official encodings of the NEO content in RDF/XML (mandatory) and in N-Triples (optional).

² This edition of the NEO Standard is available from the NSG Standards Registry, at: <http://nsgreg.nga.mil/doc/view?i=2615>.

³ RDF/XML is the mandatory encoding for OWL 2. N-Triples is an optional, plain-text format for encoding an RDF graph.

2.2 Abstract Test Suite

The abstract test suite (ATS) for the NEO Standard is specified in Annex A (normative).

3 References

3.1 Normative

The documents listed in Table 1 are indispensable to understanding and using this standard. For dated references, only the cited edition or version applies. For undated references, the latest edition or version of the referenced document (including any amendments) applies.

Table 1 – Normative References

Standard or Specification
NSG Ontology (NEO) content, encoded in technical artifacts: http://nsgreg.nga.mil/neo
ISO 19150-2:2015. <i>Geographic information – Ontology – Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)</i> : http://www.iso.org/standard/57466.html
IETF RFC 3987, <i>Internationalized Resource Identifiers (IRIs)</i> : http://www.ietf.org/rfc/rfc3987.txt
IETF RFC 4646, <i>BCP 47, Tags for Identifying Languages</i> : http://www.ietf.org/rfc/bcp/bcp47.txt
W3C OWL 2 <i>Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)</i> , 11 December 2012: http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/
W3C OWL 2 <i>Web Ontology Language: Mapping to RDF Graphs (Second Edition)</i> , 11 December 2012: http://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/
W3C <i>RDF 1.1 Concepts and Abstract Syntax</i> , 25 February 2014: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
W3C <i>RDF 1.1 N-Triples: A line-based syntax for an RDF graph</i> , 25 February 2014: http://www.w3.org/TR/n-triples/
W3C <i>rdf:PlainLiteral: A Datatype for RDF Plain Literals (Second Edition)</i> (11 December 2012): http://www.w3.org/TR/2012/REC-rdf-plain-literal-20121211/
W3C <i>SKOS Simple Knowledge Organization System</i> (18 August 2009): http://www.w3.org/TR/2009/REC-skos-reference-20090818/
W3C <i>XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes</i> (5 April 2012): http://www.w3.org/TR/xmlschema11-2/
DCMI <i>Metadata Terms</i> (14 June 2012): http://dublincore.org/documents/dcmi-terms/2
GEOINT Content Standards Board (GCSB) <i>Operations Guide</i> . NGA.SIG.0029_1.0_GCSB: http://nsgreg.nga.mil/doc/view?i=4284

3.2 Informative

The informative (non-normative) documents listed in Table 2 are useful to understanding and using this standard. For dated references, only the cited edition or version applies.

Table 2 – Informative References

Standard or Specification
ISO 19101-1:2014. <i>Geographic information – Reference model – Part 1: Fundamentals</i> (November 2014): http://www.iso.org/standard/59164.html
ISO 19103:2015. <i>Geographic information – Conceptual schema language</i> (December 2015): http://www.iso.org/standard/56734.html
ISO 19105:2000. <i>Geographic information – Conformance and testing</i> (December 2000): http://www.iso.org/standard/26010.html
ISO 19109:2015. <i>Geographic information – Rules for application schema</i> (December 2015): http://www.iso.org/standard/59193.html
ISO 19110:2016. <i>Geographic information – Methodology for feature cataloguing</i> (December 2016): http://www.iso.org/standard/57303.html
ISO 19136:2007. <i>Geographic information – Geography Markup Language (GML [version 3.2.1])</i> (September 2007): http://www.iso.org/standard/32554.html
ISO 80000-1:2009 <i>Quantities and units – Part 1: General</i> : http://www.iso.org/standard/30669.html
ISO/IEC 10646:2012, <i>Information technology – Universal Coded Character Set (UCS)</i> : http://www.iso.org/standard/56921.html
IETF RFC 1738, <i>Uniform Resource Locators (URL)</i> : http://www.ietf.org/rfc/rfc1738.txt
IETF RFC 3986, <i>Uniform Resource Identifiers (URI): Generic Syntax</i> : http://www.ietf.org/rfc/rfc3986.txt
OGC <i>Testbed-12 ShapeChange Engineering Report</i> . OGC 16-020. November 2016. http://docs.openeospatial.org/per/16-020.pdf
OMG <i>Unified Modeling Language (OMG UML), Infrastructure, Version 2.2</i> : http://www.omg.org/spec/UML/2.2/Infrastructure/PDF/
OMG <i>Unified Modeling Language (OMG UML), Superstructure, Version 2.2</i> : http://www.omg.org/spec/UML/2.2/Superstructure/PDF/
<i>National System for Geospatial Intelligence Application Schema (NAS)</i> : http://nsgreg.nga.mil/nas
<i>Shorter Oxford English Dictionary, Sixth Edition</i> (version 3.0.2.1). CD-ROM.

4 Terms, Definitions, and Acronyms

4.1 Terms and Definitions

The terms and definitions⁴ specific to this standard are presented in Table 3.

Table 3 – Definitions Applicable to this Standard

Term	Definition
annotation	An expression used to associate information with an ontology or other resource. NOTE1: An annotation is additional information associated to ontologies or ontology components that is intended for human consumption and not for use by reasoning software. NOTE2: Each annotation consists of an annotation property and an annotation value . SOURCE: OWL 2 Structural Specification (Section 3.5; Section 10)

⁴ In the definitions in Table 3, a term is styled in **bold** when the meaning of that term is specified elsewhere in the table.

Term	Definition
annotation property	A model element used to provide a textual annotation for an ontology or ontology component. SOURCE: OWL 2 Structural Specification (Section 5.5)
annotation value	A literal (including character strings), an IRI , or an anonymous individual that is the value of an annotation property . SOURCE: OWL 2 Structural Specification (Section 3.5)
axiom [OWL]	A statement of something that is true in the universe of discourse (domain). NOTE: Axioms in OWL 2 can be declarations, axioms about classes , axioms about object or data properties , datatype definitions, keys, assertions (sometimes also called <i>facts</i>), and axioms about annotations . SOURCE: OWL 2 Structural Specification (Section 9)
blank node	A node in an RDF graph that is distinct but has no IRI identifier. NOTE: A blank node cannot be referred to outside of its local graph. When stronger identification is needed, a blank node may be replaced and represented in the graph with a new, globally unique IRI (a Skolemized IRI) corresponding to the blank node. SOURCE: RDF 1.1 Concepts and Abstract Syntax (Sections 3.4, 3.5)
cardinality	The number of distinct values specified for a particular property of an individual . NOTE: Cardinality may be specified exactly (e.g., a person has exactly one biological father), or by a minimum (e.g., a parent has at least one child) and/or a maximum (e.g., in the U.S., a person may legally have at most one spouse at a time) value. SOURCE: OWL 2 Structural Specification (Section 8)
change notification (regarding a standard)	A publication in which modifications to selected items in a standard are reported in detail to the community of its users by the applicable maintenance authority. NOTE: In the NEO Standard, a change notification establishes a new content baseline . SOURCE: GCSB Operations Guide
class	A set of individuals . NOTE: In an ontology , a class typically represents a set of individuals each of which meets specified criteria for membership in the class. Class-membership criteria may be asserted formally in a class expression . SOURCE: OWL 2 Structural Specification (Section 5.1; Section 8)
class expression	A logic-based description composed from one or more classes and property expressions that represents a set of individuals (i.e., a class) by formally specifying the condition(s) on the properties of individuals belonging to the class . NOTE1: Individuals that satisfy the specified conditions are said to be instances of the class expression. SOURCE: OWL Structural Specification (Section 8)
codelist	A value domain including a code for each permissible value. SOURCE: ISO 19136:2007 (Clause 4.1.7)
concept	A mental representation of knowledge as an abstraction of the essential characteristics of a type of entity, or relationship between entities, in a subject area or domain. NOTE: Usually the abstraction is considered to be based on a generalization from experience. SOURCE: <i>The Semantic Web</i> . Michael C. Daconta, Leo J. Obrst, Kevin T. Smith. 2003.
conceptual model	A model that defines concepts of a universe of discourse . SOURCE: ISO 19101-1:2014 (Clause 4.1.5)

Term	Definition
content baseline (of a standard)	<p>The complete set of content of a standard, which is authorized (<i>i.e.</i>, 'valid') for use at a specified time.</p> <p>NOTE1: A content baseline is established by publication of a technical artifact containing the content that is valid at that time.</p> <p>NOTE2: Content baselines may be established concurrent with the publication of a new edition of a standard, or solely based on changes to the content of a standard.</p> <p>SOURCE: GCSB Operations Guide (Section 2.3.5)</p>
datatype (also: data type)	<p>An entity that refers to a set of data values.</p> <p>NOTE: A datatype is a specification of a value domain.</p> <p>EXAMPLES: Integer, Real, Decimal, Boolean, and String.</p> <p>SOURCE: OWL 2 Structural Specification (Section 4; Section 5.2)</p>
data value	<p>An element of a value domain.</p> <p>NOTE1: A data value may be used to specify an evaluated property.</p> <p>NOTE2: The set of elements of a value domain (<i>i.e.</i>, a "value space") is a datatype.</p> <p>SOURCE: OWL 2 Structural Specification (Section 4)</p>
edition (of a standard)	<p>A publication containing the entire current content of an established standard, and issued by the authorized publication authority, either as the first edition of a new standard or as a new edition (<i>i.e.</i>, revised complete version, usually numbered; for example, "2nd edition") of a previously published standard.</p> <p>SOURCE: GCSB Operations Guide</p>
entity class	<p>A modeling class that represents a feature or other geospatially-referenced information.</p> <p>SOURCE: Based on <i>entity</i>, NAS – Part 1 (Section 1.1)</p>
feature	<p>An abstraction of real-world phenomena.</p> <p>NOTE1: ISO 19101, <i>Geographic information – Reference Model</i>, defines a feature as an abstraction of real-world phenomena. Such abstractions may be represented in information systems using a variety of spatial modeling methods, including representations such as vectors, grids, and images.</p> <p>SOURCE: ISO/TC211 19101:2014 (Clause 4.1.11)</p> <p>NOTE2: The NSG Application Schema (NAS) – Part 1 also supports modeling entities that may represent other geospatially-referenced information that does not correspond to "real-world phenomena".</p>
generalization [UML]	<p>A taxonomic relationship between a more general element and a more specific element of the same element type.</p> <p>NOTE: An instance of a more specific element may be used where its more general element is allowed.</p> <p>SOURCE: ISO 19103:2015 (Clause 4.18)</p>
individual	<p>A representation of an actual object from a domain.</p> <p>SOURCE: OWL 2 Structural Specification (Section 5.6)</p> <p>NOTE: Individuals that satisfy conditions specified in a class expression are said to be instances of the class defined by that expression.</p> <p>SOURCE: OWL 2 Structural Specification (Section 8)</p>
inheritance [UML]	<p>The mechanism by which more specific classifiers incorporate structure and behavior defined by more general classifiers.</p> <p>SOURCE: ISO 19103:2015 (Clause 4.19)</p>
instance	<p>An individual that satisfies the specified conditions represented by a class expression and is therefore a member of the class represented by that expression.</p> <p>SOURCE: OWL 2 Structural Specification (Section 8)</p> <p>NOTE: Individuals represented in a data set may be called "data instances" due to being categorized, or typed, into classes used to describe the data.</p>

Term	Definition
Internationalized Resource Identifier (IRI)	<p>A sequence of characters from the Universal Character Set (Unicode/ISO 10646) [IETF RFC 3987], intended for identifying an abstract or physical resource.</p> <p>NOTE1: Every URI is by definition an IRI. A mapping from IRIs to URIs is defined, which means that IRIs can be used instead of URIs, where appropriate, to identify resources.</p> <p>SOURCE: IETF RFC 3987</p> <p>NOTE2: A resource can be anything that has identity, e.g., an OWL class or an instance of an OWL class.</p> <p>NOTE3: OWL 2 extends OWL 1 by using IRIs to identify ontologies and their elements. OWL 1 uses Uniform Resource Identifiers (URI).</p> <p>SOURCE: OWL 2 Structural Specification (Section 2.4)</p>
logical theory	<p>A set of sentences expressed in a formal language and consisting of axioms, inference rules, and theorems, which are interpreted with respect to a specified domain.</p> <p>NOTE1: A formal language is a language with identified primitive symbols and rules for constructing strings from those symbols.</p> <p>NOTE2: Axioms are sentences expressing the foundational truths of a theory; inference rules enable the derivation of valid conclusions from true sentences used as premises; and theorems are sentences that are true in the logical theory.</p> <p>SOURCES: (a) <i>The Semantic Web</i>. Michael C. Daconta, Leo J. Obrst, and Kevin T. Smith. Wiley Publishing, Inc. 2003. (b) Wikipedia: Theory (mathematical logic) (http://en.wikipedia.org/wiki/Theory_(mathematical_logic)).</p>
namespace	<p>In RDF, a common URI prefix or stem used in identifiers for a set of related resources.</p> <p>NOTE1: The RDF namespace is concatenated with the local name to create the complete URI identifier for an RDF resource.</p> <p>NOTE2: Every RDF resource is identified by a URI.</p> <p>SOURCE: ISO 19150-2:2015</p> <p>NOTE3: The NEO Standard uses the standard prefix names for namespaces as declared in the OWL Structural Specification (Section 2.4).</p>
ontology	<p>A formal representation of phenomena of a universe of discourse with an underlying vocabulary including definitions and axioms that make the intended meaning explicit and describe phenomena and their interrelationships.</p> <p>EXAMPLES: Basic Formal Ontology (BFO); Suggested Upper Merged Ontology (SUMO); Friend of a Friend (FOAF).</p> <p>NOTE: An ontology represents a universe of discourse (whether the world-at-large or a specific domain) with a formalization based in logical theory, including formally defined classes and properties, with restrictions and optionally rules, in which the structural relationships (including <i>subclass-of</i>, <i>equivalence</i>, and <i>disjoint-with</i>) are defined in a formal logic (either axiomatically or in a rule-based formulation).</p> <p>SOURCE: ISO 19150-2 citing ISO 19101-1:2014, 4.1.26</p>
property	<p>A characteristic of an entity.</p> <p>NOTE1: The characteristic describes the entity, either by a qualitative or quantitative value applicable to the entity itself, or by a relationship it has with another entity.</p> <p>NOTE2: When used to describe an entity, a property has a specified value, either a data value or another entity to which it is related in the indicated way.</p> <p>SOURCE: Based on OWL 2 Structural Specification</p>
property (datatype) [OWL]	<p>A property whose data value is a literal.</p> <p>NOTE1: Formally, a literal consists of a string (<i>i.e.</i>, lexical form) and a datatype; the string denotes a value in the range of the lexical space of the associated datatype.</p> <p>NOTE2: Also see annotation property.</p> <p>SOURCE: OWL 2 Structural Specification (5.4; (NOTE1) 5.7)</p>

Term	Definition
property (object) [OWL]	A property that is a relationship between two individuals . SOURCE: OWL 2 Structural Specification and Functional-Style Syntax (5.3)
subclass	A relationship between two classes such that each instance of the more specific class is also an instance of the more general class . NOTE1: In OWL, an axiom expressed with “SubClassOf” states that each instance of one class is also an instance of another, more general class . NOTE2: Subclass axioms may be used to construct a hierarchy of classes . SOURCE: OWL Structural Specification (Section 9.1)
URI Base	A base URI in a domain owned by the organization that maintains the model or ontology . SOURCE: ISO 19150-2:2015 (Clause 6.2.2)
Uniform Resource Identifier (URI)	A compact string of characters for identifying an abstract or physical resource. NOTE1: A resource can be anything that has identity; for example, an OWL class , or an individual that is an instance of an OWL class . NOTE2: A URI identifies a resource either by location, or by name, or both. NOTE3: URIs are limited to a subset of the ASCII character set. SOURCE: IETF RFC 3986
Uniform Resource Locator (URL)	A compact string representation for location and access of a resource available on the internet. NOTE: A URL is a type of URI . SOURCE: IETF RFC1738
universe of discourse	A view of the real or hypothetical world that includes everything of interest. SOURCE: ISO 19150-2:2015 citing ISO 19101-1:2014 (Clause 4.1.38)

4.2 Acronyms

The acronyms that are used in this standard are specified in the following list.

ASCII	American Standard Code for Information Interchange
ATS	Abstract Test Suite
API	Application Programming Interface
BCP	Best Current Practice
DCMI	Dublin Core Metadata Initiative
DoD	(U.S.) Department of Defense
GCSB	GEOINT Content Standards Board
GEOINT	Geospatial Intelligence
GFM	General Feature Model
IANA	Internet Assigned Numbers Authority
IC	(U.S.) Intelligence Community
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
IUT	Implementation Under Test
NAS	NSG Application Schema
NGA	National Geospatial-Intelligence Agency
NSG	National System for Geospatial Intelligence
NEO	NSG Enterprise Ontology
OGC	Open Geospatial Consortium
OMG	Object Management Group
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
REST	REpresentational State Transfer

SDO	Standards Development Organization
SKOS	Simple Knowledge Organization System
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language

4.3 Presentation Font

The general text of this document is presented using Arial font. Encoding elements for the information model are presented using Courier New font (e.g., `owl:Class`).

5 Ontology Specification

5.1 Introduction

The NEO Standard establishes the terminological, semantic, and structural basis for specifying a conceptual model in the form of a logical theory that defines domain concepts used in Geospatial Intelligence (GEOINT) information shared in the NSG. This section on ontology specification defines the information model for representing the NEO content and specifies two encodings for that content.

The NEO information model is based on the General Feature Model (GFM) of ISO 19109:2015. The formal representation for concepts in the NEO information model is based on the family of W3C Recommendations (i.e., standards) defining OWL 2. The representation of the NEO model using OWL 2 classes and properties supports the use of NEO content to describe data exchanged among automated information systems in a machine-processable way.⁵

To support usage on the Web, the NEO Standard prescribes Internationalized Resource Identifiers (IRIs) to identify the NEO content. To support applications that require a Web encoding format, the NEO Standard specifies two W3C encodings: RDF/XML and N-Triples. The encoded NEO content is published in the NSG-unique Standards Register of the NSG Standards Registry and is also accessible through the REST API component of the NSG Standards Registry.

The ontology specification in the NEO Standard defines the NEO content in three ways, by providing:

1. Content specification – the information model for NEO entity classes (types), their properties (attributes and relationships), and datatypes;
2. Content identification – IRIs for unique identification of items in the NEO content; and
3. Content encoding – Specifications for the RDF/XML and N-Triples encodings of the NEO content.

Section 5.2 specifies the information model for the NEO in a diagram together with tabular specifications for all included modeling elements.

Section 5.3 specifies the use of OWL 2 to represent elements of the information model.

Section 5.4 specifies the implementation of the NEO content in two supported encodings: (1) RDF/XML and (2) N-Triples. This allows instance data in either RDF/XML or N-Triples formats to be related to NEO concepts in order to provide semantics for data exchanged among automated information systems.

⁵ *Data on the Web Best Practices, a W3C Recommendation* (31 January 2017). Available online at: <http://www.w3.org/TR/dwbp/>.

NOTE The NEO Standard document does not include the specific ontology content (*i.e.*, specific entity classes) included in any particular NEO content baseline. NEO content is published in technical artifacts (encodings) that may be accessed online in the NSG-unique Standards Register of the NSG Standards Registry. NEO content is also accessible through the REST API component of the NSG Standards Registry. See Section 6.3.

5.2 *NEO Information Model*

5.2.1 Introduction

The information model for NEO defines the modeling concepts needed to represent a logical theory of entity types and properties (including relationships) used for the description of geospatial information in the NSG.

The NEO information model is based on the General Feature Model (GFM) defined in ISO 19109 for use in ISO geographic-information standards. The GFM provides “a model of the concepts required to classify a view of the real world” in which the universe of discourse is described using feature types with properties including attributes and associations. (ISO 19109, 7.4.2) The GFM is a metamodel for defining features, and thus provides a framework for the definition of domain concepts used in the description of GEOINT data. The NEO information model additionally specifies documentation properties to record the semantics and provenance of NEO content.

The NEO information model includes the following kinds of information-modeling constructs:

- ontology class (for representing and documenting the NEO content),
- entity class (for representing object types and data types included in the NEO content),
- class axiom for representing the disjointness constraint on collections of sibling subclasses,
- entity property (for representing the characteristics of entities, including relationships), and
- a set of documentation properties (for information about the ontology and its components).

Section 5.2.2 presents a diagram of the information model (Figure 1) and explains the structure of the tabular specifications for NEO modeling concepts (Sections 5.2.3. through 5.2.7).⁶

Section 5.2.8 specifies the datatypes used in the NEO information model (Figure 2).

⁶ See Annex E for explanation of the notation.

5.2.2 Diagram of the Information Model

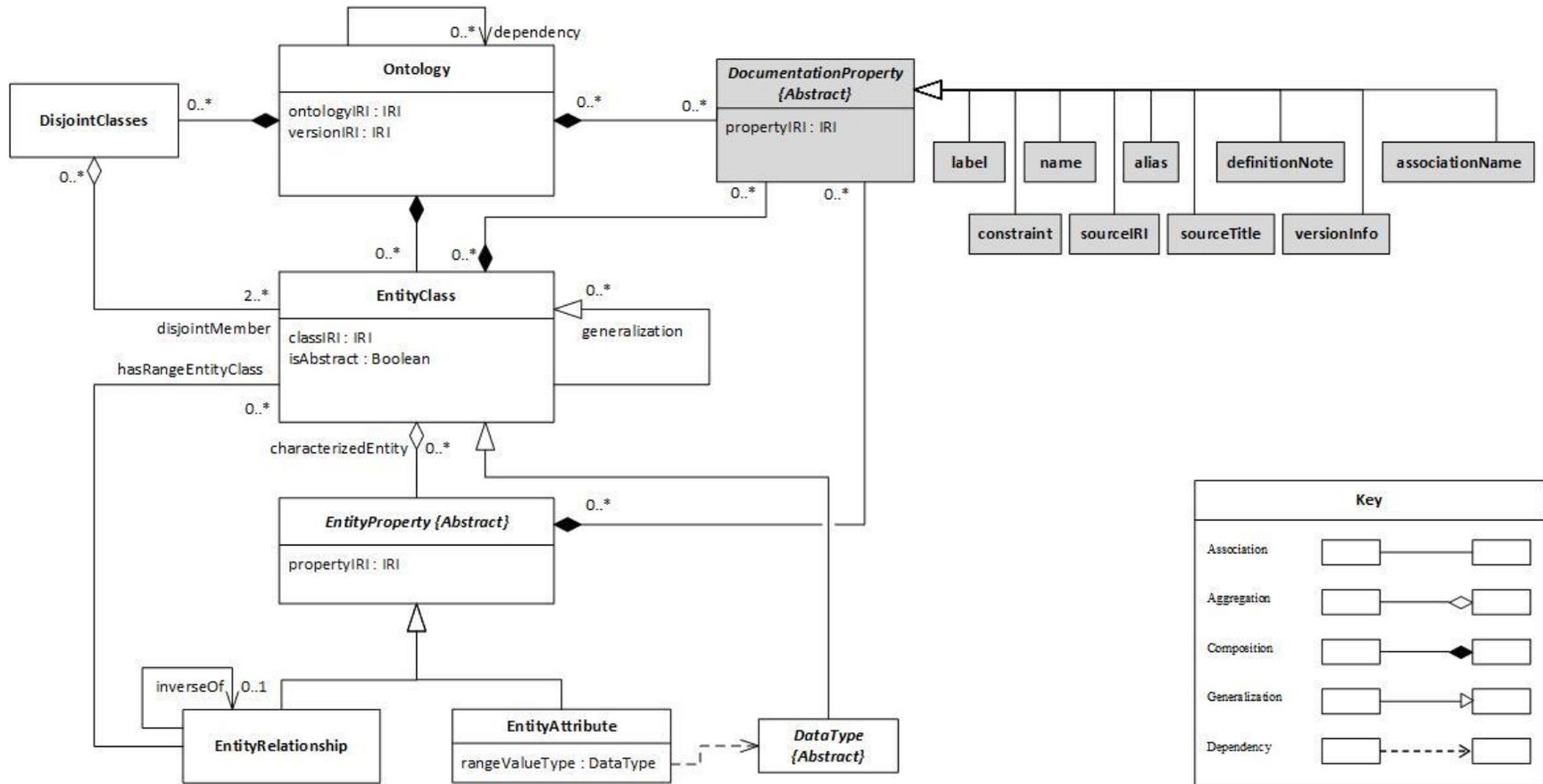


Figure 1 – Overview of the NEO Information Model

Each modeling concept in the NEO information model is defined, together with its properties, in tables in the following sections. The table format used to document these model components is as follows:

- The **Reference** column consists of a sequentially-assigned, non-normative identifier of the element (class or property) that is provided for cross-referencing purposes. It may vary from edition-to-edition of this document.
- The **NEO Modeling Concept** column specifies the class name or property name for the information modeling concept. Properties are either attributes or relationships. Relationship names are prefixed by the italicized phrase “*Role name*”.
 - A specified class in the model has a name and always appears in the table on a light-grey highlighted row above its properties.
 - The properties (attributes and/or relationships) of a model class are specified in subsequent rows of the table below the class row.
- The **Definition** column specifies the definition of the model class or property.
- The **Source of Definition** column records the source of the definition of the information modeling concept if it is based on a definition in an external source. If blank, the source of the definition is this standard.
- The **Obligation** column specifies if the property is **Mandatory**, Conditional, or *Optional*. This column has values only for properties and is dark-grey filled on rows containing classes.
 - Properties whose obligation is “**Mandatory**” shall be populated in accordance with the property definition and any associated guidance.
 - Properties whose obligation is “Conditional” are mandatory when the stated condition is satisfied, in which case they shall be populated in accordance with the property definition and any associated guidance.
 - Properties whose obligation is “*Optional*” are optional, but their population is good business practice when the applicable information is available.
- The **Multiplicity** column indicates the number of instances of the value type of the property that are permitted by this information model. In the case that more than a single domain value of the property is allowed, an indication may also be included in this column if the ordering of the domain values is significant. This column has values only for properties and is dark-grey filled on rows containing classes.
- The **Value Type** column indicates the modeling concept or datatype that is used to define the value(s) of the property. This column has values only for properties and is dark-grey filled on rows containing classes.

5.2.3 Ontology

The Ontology model component represents a logical theory as a resource containing component classes, properties, and axioms. The ontology is assigned a unique identifier in the form of an IRI. An ontology may be formally related to one or more other ontologies whose content is imported. The specification of Ontology and its properties is presented in Table 4. In encodings, an Ontology shall also be described using the applicable documentation properties, including its name and source information (see Section 5.2.7).

Table 4 – Definition of Ontology and its Properties

Ref #	NEO Modeling Concept	Definition	Source of Definition	Obligation	Multiplicity	Value Type
1	Ontology	A formal representation of phenomena of a universe of discourse with an underlying vocabulary including definitions and axioms that make the intended meaning explicit and describe phenomena and their interrelationships.	ISO 19101-1:2014, 4.1.26			
2	ontologyIRI	A uniform resource identifier (URI) that uniquely identifies an ontology, consisting of a URI base owned by the organization that maintains the ontology, concatenated (following a separator "/") with an abbreviation of the name of the ontology.	ISO 19150-2:2015, 6.2.2	Mandatory	Exactly one	IRI
3	versionIRI	A uniform resource identifier (URI) that uniquely identifies a specific version of an ontology, consisting of a URI base owned by the organization that maintains the ontology, concatenated (following a separator "/") with an abbreviation of the name of the ontology and (following a separator "/") with the version indicator (e.g., year or version number).	ISO 19150-2:2015, 6.3.3	Mandatory	Exactly one	IRI
4	<i>Role name:</i> dependency	An ontology whose content is imported into this ontology.	ISO 19150-2:2015, 6.3.3	<i>Optional</i>	Zero or more	Ontology

5.2.4 EntityClass

The EntityClass model component is used to define the entity types in the ontology. An entity type is a set of individuals that share the same nature and typical characteristics. The specification of EntityClass with its properties is presented in Table 5. Each EntityClass in the NEO content is assigned a unique identifier in the form of an IRI. Each EntityClass may be defined with specific properties (*i.e.*, attributes and relationships) that characterize individuals of that type. In encodings, each EntityClass will also be described using the applicable documentation properties, including its name, source reference, and applicable constraints (see Section 5.2.7).

Table 5 – Definition of EntityClass and its Properties

Ref #	NEO Modeling Concept	Definition	Source of Definition	Obligation	Multiplicity	Value Type
1	EntityClass	A concept for a set of individuals that share the same nature and specific properties.	Based on UML 2.4			
2	classIRI	A uniform resource identifier (URI) that uniquely identifies a class, consisting of the ontology IRI concatenated (following a separator) with the label of the class.	ISO 19150-2:2015, 6.2.4	Mandatory	Exactly one	IRI
3	isAbstract	A Boolean value indicating that this entity class is abstract (<i>i.e.</i> , not intended to be used directly in classification of data instances).	ISO 19109:2015, 7.4.5 isAbstract	Conditional	If applicable, then exactly one.	Boolean
4	<i>Role name:</i> generalization	The relationship between an entity class and its superclass(es), such that all individuals belonging to the subclass also belong to the superclass and satisfy its definition. A subclass inherits the properties of its superclass(es). The superclass is the generalized class, while the subclasses are typically specified with additional properties.	ISO 19109:2015, 7.4.5 superType; 7.4.12 generalization	Conditional	If applicable, then one or more.	EntityClass

Section 5.2.6 describes a special use of the EntityClass model component to define a relationship that has attributes.

5.2.5 DisjointClasses Axiom

The DisjointClasses model component represents sets of entity classes which shall not have any individuals as members in common.

Table 6 – Definition of DisjointClasses

Ref #	NEO Modeling Concept	Definition	Source of Definition	Obligation	Multiplicity	Value Type
1	DisjointClasses	A collection of entity classes, all of which are pairwise disjoint, indicating that no individual can belong at the same time to more than one of the member classes ("types"). For example, a collection of sibling subclasses having the same generalization, where instances of the supertype shall not be an instance of more than one of the subtypes.	ISO 19109:2015, 7.4.12 uniqueInstance			
2	<i>Role name:</i> disjointMember	An entity class belonging to this collection of disjoint classes.		Conditional	If applicable, then two or more.	EntityClass

5.2.6 EntityProperty and its Subclasses

The EntityProperty model component is an abstract generalization for the representation of attributes and relationships that characterize the instances of an EntityClass. Each property is assigned a unique identifier in the form of an IRI. The specification of EntityProperty and its properties is presented in Table 7.

EntityProperty has two concrete subclasses, which are specialized for the representation of attributes (data-valued characteristics of an entity) and relationships (associations between individuals). The subclasses of EntityProperty are presented in Table 8 and Table 9. Every EntityAttribute and EntityRelationship will have attribution inherited from this abstract superclass, in addition to the properties defined for the specialization. Each EntityAttribute and EntityRelationship will also be described using the applicable documentation properties, including its name and source reference (see Section 5.2.7).

Table 7 – Definition of EntityProperty and its Properties

Ref #	NEO Modeling Concept	Definition	Source of Definition	Obligation	Multiplicity	Value Type
1	<i>EntityProperty</i> { <i>Abstract</i> }	A concept for a characteristic of an entity (either an attribute of the entity or a relationship to another entity).	Based on UML 2.4			
2	propertyIRI	A uniform resource identifier (URI) that uniquely identifies a property, consisting of the ontology IRI concatenated (following a separator) with the label of the property.	ISO 19150-2:2015, 6.2.6 (Table 7)	Mandatory	Exactly one	IRI
3	<i>Role name:</i> characterizedEntity	The relationship between an entity property and an entity class whose individual members may be described using this property.		<i>Optional</i>	Zero or more	EntityClass

The EntityAttribute model component represents a characteristic that describes an entity in terms of a data value. The data type may be simple or complex. In addition to the properties listed below, EntityAttribute also inherits the properties of EntityProperty specified in Table 7.

Table 8 – Definition of EntityAttribute and its Properties

Ref #	NEO Modeling Concept	Definition	Source of Definition	Obligation	Multiplicity	Value Type
1	EntityAttribute	A concept that represents a characteristic that describes an entity in terms of a data value, data without reference to another entity.				
2	rangeValueType	The datatype for all values of this entity attribute.		Mandatory	Exactly one	DataType (see Section 5.2.8)

The EntityRelationship model component represents a characteristic that describes an entity in terms of its association with another entity. An EntityRelationship may have an inverse; that is, it may be paired with another EntityRelationship specifying a relationship in the reverse direction from the original value-type (range) entity to the original domain. The meanings of an EntityRelationship and its inverse are related but not necessarily the same; each direction may have a distinct meaning. An EntityRelationship may be derived from a named association between two entity classes; in that case, the name of the association is recorded on the EntityRelationship using a documentation property (associationName; see Section 5.2.7). In addition to the properties listed below, EntityRelationship also inherits the properties of EntityProperty specified in Table 7.

Table 9 – Definition of EntityRelationship and its Properties

Ref #	NEO Modeling Concept	Definition	Source of Definition	Obligation	Multiplicity	Value Type
1	EntityRelationship	A concept that represents a relationship between two individuals.				
2	<i>Role name:</i> hasRangeEntityClass	The entity class that is the value type (<i>i.e.</i> , range) for this entity relationship.		<i>Optional</i>	Zero or more	EntityClass
3	<i>Role name:</i> inverseOf	An entity relationship that relates the same two entities as this relationship but with the opposite directionality (<i>i.e.</i> , interchanging the domain and range).		Conditional	If applicable, then exactly one.	EntityRelationship

The EntityRelationship model component is not used to represent a relationship which itself has properties (corresponding to the UML construct for an association class). The NEO information model does not include a specialized modeling element for that concept. Instead, NEO models relationships-with-properties by using an EntityClass to represent the relationship and using two roles (hasRangeEntityClass) to relate the EntityClass to each of the two individuals associated by the relationship-with-properties. Those individuals, in turn, have inverse hasRangeEntityClass roles that relate them to the EntityClass representing the relationship between them.⁷

⁷ See the informative reference (Section 3.2) *OGC Testbed-12 ShapeChange Engineering Report* (section 8.2.5) for an explanation (with diagrams) of the pattern of entity classes and relationships that is used in the NEO content to represent content derived from UML association classes. The approach is based on GML 3.3 (see https://portal.opengeospatial.org/files/?artifact_id=46568), Section 12.3.

5.2.7 DocumentationProperty

The model component `DocumentationProperty` is the set of properties that represent labels, provenance information, and other annotations used to describe the ontology and ontology components, where applicable. The documentation properties are defined as follows, including the model components to which they apply:

- **label** – A compressed human-readable designator for a resource that may be used as the terminal segment of its resource IRI. The language may be indicated. [Source: Based on ISO 19135:2005, 7.2 Item identifiers; ISO 19150-2:2015, 6.3.3 and 6.4.3]
 - A label is mandatory on the ontology and all ontology components.
- **name** – The preferred human-readable designator that is used to denote the concept in a specified language. [Source: ISO 19135:2005, 7.2 Item names, 8.6.2 name]
 - A name is mandatory on the ontology and all ontology components.
- **alias** – A functionally equivalent synonym for a concept in an alternative context or language. [Source: ISO 19110:2016, Table B.2, No. 2.5 aliases]
 - Any concept may have zero or more aliases.
- **definitionNote** – A precise statement of the nature and specific properties of a concept, followed by an optional statement about relevant non-essential qualities, variations, scope, and/or examples. The language of the definition may be indicated. [Source: ISO 19135:2005, 7.3 Definitions; ISO 19150-2:2015, 6.4.3]
 - A definitionNote is mandatory on the ontology and all ontology components.
- **associationName** – The name of the (UML) association from which an entity relationship representing an association role was derived. [Source: ISO 19150-2:2015, 6.10.2.3 Rules]
 - Each entity relationship derived from an association role of a named association is annotated with the association name.
- **constraint** – A description of a condition or restriction used for declaring some of the semantics of an entity class. [Source: ISO 19150-2:2015, 6.11 Constraint]
 - A constraint may be used only on an entity class.
- **sourceIRI** – The Internationalized Resource Identifier (IRI) of the recommended reference to be used for information about the concept. [Source: ISO 19115:2003, 2.4.2.3 (line 96), sourceCitation]
 - A sourceIRI is mandatory on the ontology and all ontology components.
- **sourceTitle** – The title of the ontology reference document or standard on which the ontology is based. [Source: ISO 19150-2:2015, 6.3.3]
 - A sourceTitle is mandatory on the ontology (and excluded on ontology components).
- **versionInfo** – A character string indicating a unique state in the life of a managed resource (for example: by date or number). [Source: Based on ISO 19135:2005]
 - A versionInfo annotation is mandatory on the ontology (and excluded on ontology components).

5.2.8 DataType

5.2.8.1 Introduction

The NEO information model specifies datatypes for use in defining the ranges of properties included in the ontology. NEO datatypes are classes that define allowed types of data values. Distinctions among NEO datatypes are based primarily on structural characteristics. Some datatypes (for example, those classified under `MeasureDatatype`) also have a common semantics.

The NEO information model includes the following top-level classes of datatypes:

- Primitive Datatypes, which consist of a simple data value that is not decomposable into other datatypes (Sections 5.2.8.2 through 5.2.8.12);
- Measure Datatypes, which specify a numeric value accompanied by a unit of measure (Sections 5.2.8.13 and 5.2.8.14);

- Enumerated Types, which specify a set of data values that may be used as the value of a property (Sections 5.2.8.15, 5.2.8.16, and 5.2.8.17);
- Complex Datatypes, which specify compositions consisting of multiple properties, including some that represent metadata about an evaluated property (Sections 5.2.8.18, 5.2.8.19, and 5.2.8.20).

The NEO information model datatype hierarchy also includes the abstract class `DataComponent`, one of a set of NAS concepts for dynamically specifying externally-defined properties and their value types.

Figure 2 presents a diagram of the datatypes included in the NEO information model.⁸ Each non-abstract datatype is described in the sections following the diagram.

⁸ The NEO content includes numerous additional datatypes that are specializations of the datatypes in the NEO information model presented here (e.g., `Binary` (a Primitive Datatype); `Real Interval` and `Currency Value` (Complex Datatypes); and the many types of `DatatypeUnion` and `DatatypeMeta`, such as `MaritimeBottomCharactermaterialCodeReason` and `AerialTypeCodeMeta`).

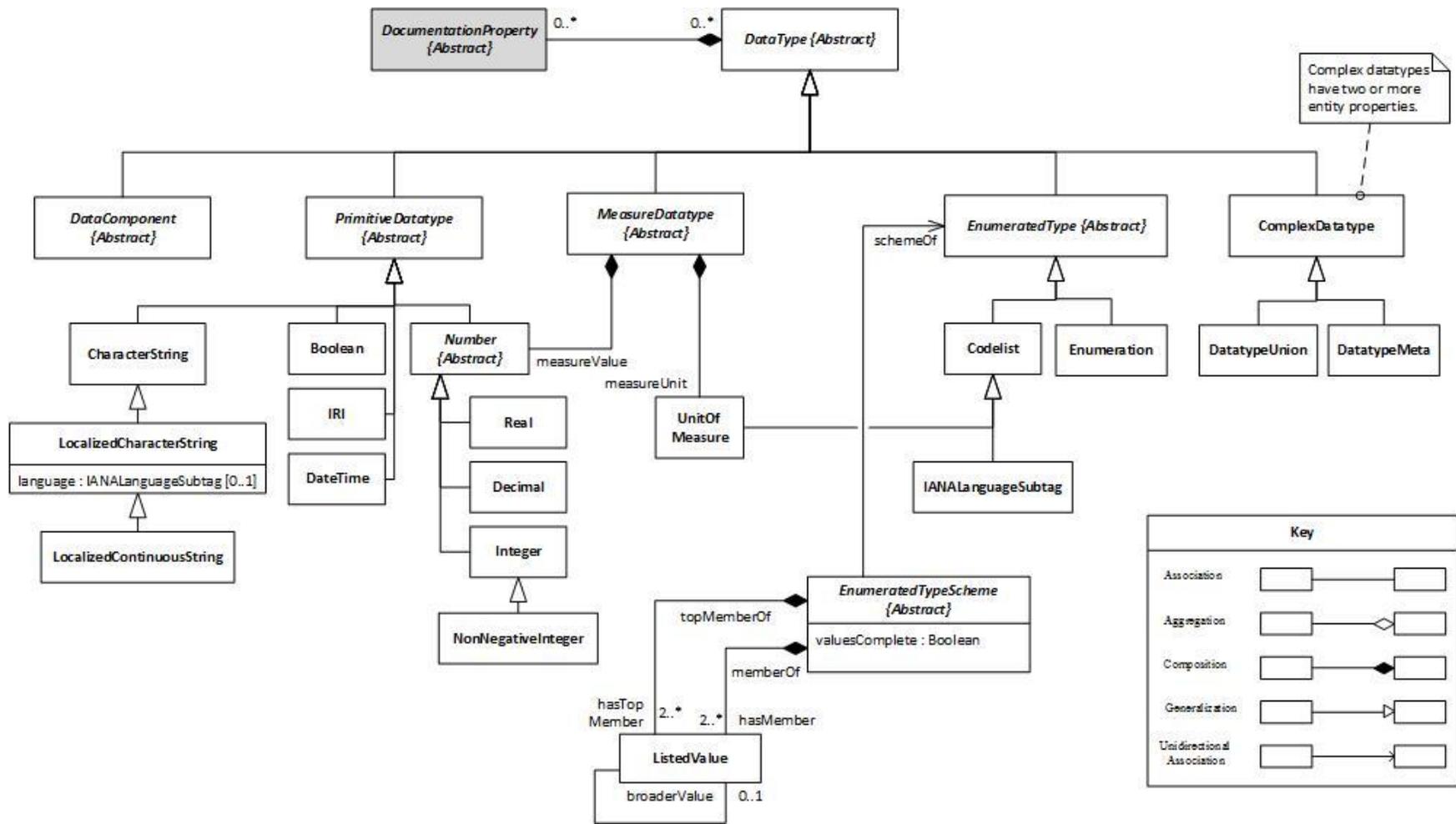


Figure 2 – Model of the NEO Datatypes

5.2.8.2 CharacterString

The datatype `CharacterString` represents a finite-length sequence of zero or more characters from the Universal Character Set (Unicode), as specified by ISO/IEC 10646. The character string may be accompanied by a formal identifier (*i.e.*, a token or “tag”) used to identify the natural language of the expression represented by the string. A character string may be further specified, including with respect to length (exact, minimum, or maximum) or pattern (for example, a pattern that excludes space characters). The string value itself is simple; that is, it cannot be decomposed further into other datatypes.

5.2.8.3 LocalizedCharacterString

The datatype `LocalizedCharacterString` represents a character string for which the natural language to use in interpreting the content is specified by a language code (“tag”). A character string is a finite-length sequence of zero or more characters from the Universal Character Set (Unicode). A language tag is a lowercase abbreviation for the natural language of the expression represented by a character string.

An attribute of `LocalizedCharacterString` uses the codelist `IANALanguageSubtag` to indicate the language of a character string. The `IANALanguageSubtag` codelist represents a set of formal identifiers for natural languages, as defined by BCP 47 (currently represented by RFC 4646 and RFC 4647) or its successor(s). IANA language subtags are the lowercase two-character codes contained in the Language Subtag registry administered by the Internet Assigned Numbers Authority (IANA) in accordance with the Internet Engineering Task Force (IETF) Recommendation for Comment (RFC) 5646.

The language codes in the IANA Language Subtag registry are based on the International Organization for Standardization (ISO) 639 series of standards.⁹

5.2.8.4 LocalizedContinuousString

The datatype `LocalizedContinuousString` represents a character string having no whitespace characters (unless they are encoded by ‘%20’) and for which the natural language to use in interpreting the content is specified by a language code (“tag”).

5.2.8.5 Boolean

The datatype `Boolean` represents the values of a two-valued logic. A Boolean value is either `TRUE` or `FALSE`.

5.2.8.6 IRI

The datatype `IRI` represents International Resource Identifiers (IRIs). An IRI is a finite-length sequence of characters from the Universal Character Set (Unicode/ISO 10646) that is intended to identify an abstract or physical resource as described in IETF RFC 3987.

5.2.8.7 DateTime

The datatype `DateTime` represents the set of specialized character strings consisting of digits with leading zeroes that contain values for century (CC), year (YY), month (MM), day (DD), and hours (hh), minutes (mm), and seconds (ss), with a timezone offset from Coordinated Universal Time (UTC) or ‘Z’ for UTC, formatted in accordance with IETF RFC 3339, which is ‘CCYY-MM-DDThh:mm:ssZ’ (for example: ‘1985-04-12T11:45:20Z’ for 11 hours, 45 minutes and 20 seconds UTC on 12 April 1985). The format conforms to `xsd:dateTime`¹⁰. Midnight is understood to be 00:00:00 (the beginning of a day); when the time is not specified then midnight in the local time zone is typically implied.

5.2.8.8 Number

The datatype `Number` is the generalization of specific primitive datatypes (for example: `Real` and `Integer`). The `Number` datatypes represent quantitative values that can be used to specify the numeric amount of a `MeasureDatatype`.

⁹ The complete IANA Language Subtag registry content is available at the following URL: <http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry>.

¹⁰ See <http://www.w3.org/TR/xmlschema-2/#dateTime>.

5.2.8.9 Real

The datatype Real represents a signed real (floating point) number consisting of a mantissa and an exponent, which represents a value to a precision given by the number of digits shown, but is not necessarily the exact value.

5.2.8.10 Decimal

The datatype Decimal represents a number from the subset of real numbers that can be obtained by multiplying an integer by a non-positive power of ten, *i.e.*, expressible as $i \times 10^{-n}$ where i and n are integers and $n \geq 0$.

5.2.8.11 Integer

The datatype Integer represents a signed integer number, as an exact integer value with no fractional part.

5.2.8.12 NonNegativeInteger

The datatype NonNegativeInteger represents an integer domain value that is restricted to be non-negative (*i.e.*, either zero or positive).

5.2.8.13 MeasureDatatype

The datatype MeasureDatatype represents the set of data values that represent a quantity as a numeric amount expressed in terms of a specific unit based on a scale or using a scalar reference system.¹¹ This datatype is used to represent data values that are physical quantities.

5.2.8.14 UnitOfMeasure

The datatype UnitOfMeasure is the class of defined units of measure for physical quantities. Values of this datatype are used to specify the numeric amount of a measure in a commonly agreed scale. Units of measure used in the NEO content are defined by the ISO 80000 (multi-part) standard.

5.2.8.15 Enumeration (Non-extensible EnumeratedType)

Enumerated datatypes are either enumerations (closed domains for which all values have been represented) or codelists (open domains which are potentially incomplete and may have new values).

Each EnumeratedType is associated with an EnumeratedTypeScheme that models the listed values as members of a collection of related data values which may be hierarchically structured, with broader values and “top members”.

While the EnumeratedType and EnumeratedTypeScheme for a specific concept differ in formal structure, the meaning of the underlying domain concept is the same (for example, the EnumeratedType and EnumeratedTypeScheme for BuildingFeatureFunction are different ways of organizing “the allowed set of functions that may be identified for buildings”).

The datatype Enumeration represents a set of related domain values (called listed values) that are allowable values for a property. The listed values of an enumeration are completely specified, and an enumeration is not extensible.

5.2.8.16 Codelist (Extensible EnumeratedType)

The class Codelist represents a set of related domain values that are allowable values for a property. The listed values of a codelist are not considered as completely specified, and a codelist is therefore extensible by following the applicable governance procedures for that codelist.

Each Codelist is associated with an EnumeratedTypeScheme (Section 5.2.8.15).

5.2.8.17 ListedValue

The datatype ListedValue represents items that are specified as a member of an enumerated type.

5.2.8.18 ComplexDatatype

The datatype ComplexDatatype represents datatypes consisting of multiple properties. At least one of the component properties provides a principal data value, that is, a data value characterizing a domain entity (for example: the

¹¹ *OpenGIS Geography Markup Language (GML) Encoding Standard, v3.2.1* (ref # OGC 07-036). Ed., Clemens Portele. Open Geospatial Consortium Inc. 2007-08-27. When used as a noun, 'measure' is a synonym for physical quantity.

elevation of an aerodrome, or the contact address for a facility), while the other component properties may provide contextual information or metadata about the asserted domain value (for example: an estimated accuracy for the elevation measure, or a reason why the contact address is absent).

For example, a complex datatype may combine an elevation measure with identification of the reference datum and an evaluation of its accuracy. The complex datatype would be composed of three properties each having its own datatype: the first property specifying a real value for the elevation, the second property specifying a vertical reference datum for the elevation measure, and a third property specifying the accuracy of the elevation measure.

Other complex datatypes are used to represent intervals of various kinds, including intervals of real numbers and time intervals. A complex datatype for an interval includes properties for the beginning and end points of the interval, and an indication of whether each interval limit is open, closed, or inapplicable.

Two special patterns of complex datatypes, `DatatypeMeta` and `DatatypeUnion`, are described in the following sections.

5.2.8.19 `DatatypeUnion`

The datatype `DatatypeUnion` represents a complex datatype consisting of a set of properties that are alternatives. Only one of the constituent properties is evaluated for any data instance.¹²

One pattern for a datatype union consists of alternative properties that either provide a value for a domain attribute or else a reason that the data value is absent. For example, a datatype union may be used for reporting either the availability status of an aerodrome runway, or the meaning of reporting no value (for example: "No Information").

5.2.8.20 `DatatypeMeta`

The datatype `DatatypeMeta` represents a datatype with at least one property that provides a principal data value optionally accompanied by property-level metadata including, for example, restrictions on use, temporal extent, or provenance of the principal data value. Each metadata component has its own datatype. The included datatypes may be primitive or complex.

5.3 *NEO Representation using Semantic Web Languages*

5.3.1 The Semantic Web

The information modeling elements of the NEO information model can be represented using the Web Ontology Language, Second Edition (OWL 2), as defined by W3C Recommendations for the Semantic Web. The use of OWL 2 enables the encoding of NEO content in machine-readable formats that can be used with Web-based applications and shared on the Web to provide machine-processable semantics for data published from multiple sources (for example, as Linked Data).¹³

The Semantic Web is a virtual set of distributed data accessible on the internet that is represented using standards-based, machine-processable descriptions that allow the data to be application-independent and available for re-use in accordance with a framework of common standards.¹⁴ Data in the Semantic Web can be discovered, queried, aggregated, and analyzed as part of the larger information ecosystem by leveraging the semantics (*i.e.*, meanings) of the data. The phrase "the Web of Data" is used synonymously with the Semantic Web in this sense.¹⁵

The term "Semantic Web" also encompasses the technologies, including the standards and operational infrastructure, that support the creation of the Web of Data. Semantic Web standards define a framework (including representation languages and exchange formats) for describing data in a reusable, machine-processable way, as well as guidelines for creating the operational environment on the Web.¹⁶ Finally, the Semantic Web relies on an implemented technology infrastructure that enables the real-time publication, linking, and processing of data published in Semantic Web exchange formats.

The World Wide Web Consortium (W3C) has developed a set of recommendations (standards) that can be used together with non-W3C standards (such as Unicode) to support the representation and exchange of information on the Web. All of these recommendations depend upon unique identifiers using standardized character sets to identify

¹² In UML terms, this datatype has the <<union>> stereotype pattern.

¹³ A similar approach to data definition and data linking may be used in a closed networked system, rather than on the open internet, when required for mission-specific purposes.

¹⁴ *The Semantic Web*. Michael C. Daconta, Leo J. Obrst, and Kevin T. Smith. Wiley Publishing, Inc. 2003. Page 4.

¹⁵ Linked Data Glossary, W3C Working Group Note 27 June 2013 (<http://www.w3.org/TR/ld-glossary/>)

¹⁶ "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries." (W3C FAQ, *What is the Semantic Web*: <http://www.w3.org/2001/sw/SW-FAQ>)

resources on the Web. Initial efforts to support the sharing of information on the Web through the creation of Extensible Markup Language (XML)¹⁷ focused on the representation of individuals as data objects (*i.e.*, instance data) together with metadata about them. Subsequently, three W3C recommendations (RDF, RDFS, and OWL) defined new representation languages that are used to formalize the semantics of data and its real-world domains in a machine-processable form. Those recommendations together enable support for automated logical reasoning about the data (such as inferencing to check constraints or to conclude additional facts from known data) as well as querying for data across the Web.

The set of standards used to enable sharing of the semantics of information on the Web is often referred to as the “Semantic Web Stack”, because later recommendations built upon and extended the capabilities of earlier standards. Figure 3 shows graphically the reliance on and dependencies among the recommendations and standards that are used together to enable the Semantic Web.

The key standards for capturing the semantics of data are RDF, RDFS, and OWL. The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent complex knowledge about entities, groups of entities, and relationships between entities. OWL is a formal language based on computational logic, which allows knowledge expressed in OWL to be processed by computer programs, including reasoners.¹⁸

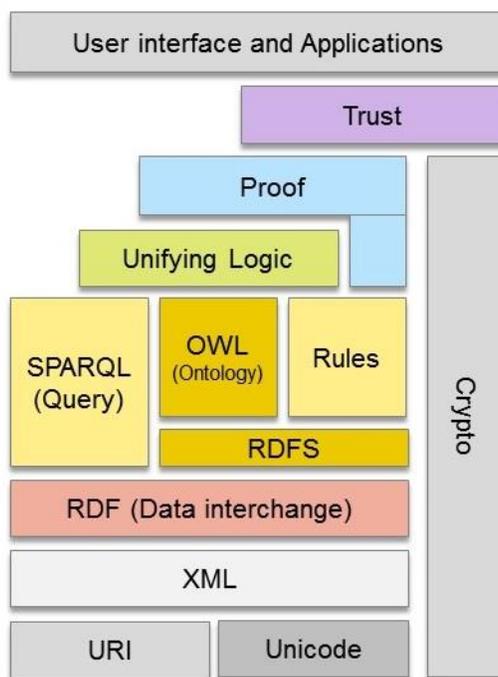


Figure 3 – The Semantic Web Stack¹⁹

The Semantic Web stack supports a common framework that allows data to be shared and reused across application, enterprise, and community boundaries, through the extraction, representation, storage, retrieval, and analysis of machine-processable data.²⁰ This framework includes standards that define exchange formats (primarily RDF) for sharing data on the Web (sometimes called the “Web of Data”).

Data that is prepared, published, shared, and consumed in the Semantic Web is called “Linked Data”. As defined by the W3C, Linked Data is data that is implemented using:

A pattern for hyperlinking machine-readable data sets to each other using Semantic Web techniques, especially via the use of RDF and URIs. This enables distributed SPARQL queries of the data sets and a browsing or discovery approach to finding information (as compared to a search strategy). Linked Data is

¹⁷ XML was developed in the late 1990s to provide a syntax for creating markup languages to capture metadata.

¹⁸ W3C Web Ontology Language (OWL). Accessed online at: <http://www.w3.org/OWL/>.

¹⁹ Based on *Semantic Web and Other W3C Technologies to Watch*. Steve Bratt, CEO, W3C. October 2006. Retrieved online at: <http://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/>. There is some variation in depictions of the stack, which has changed over the years with the addition of new recommendations such as the Rules Interchange Format (RIF) and the adoption of IRIs to provide a broader method than URIs for constructing unique identifiers. There are also variations in which the diagram consists of uniform flat layers, although in fact the relationship between layers is more complex than in the stack-of-pancake depictions.

²⁰ Linked Data Glossary, W3C Working Group Note 27 June 2013. (<http://www.w3.org/TR/ld-glossary/>)

intended for access by both humans and machines. Linked Data uses the RDF family of standards for data interchange (e.g., RDF/XML, RDFa, Turtle) and query (SPARQL).²¹

Guidance on the production and publication of data instances is outside the scope of the NEO Standard. In January 2017, the W3C Data Activity published “Data on the Web Best Practices”, a W3C Recommendation (31 January 2017), available at <http://www.w3.org/TR/dwbp/>. Topics covered in that publication include: metadata, licenses, provenance, quality, versioning, identifiers, formats, vocabularies, access (APIs), and data preservation.²²

5.3.2 Selecting OWL Constructs for Representation of NEO Content

The formal representation language used for the NEO content is the W3C Web Ontology Language defined in the *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*.²³ Elements from OWL 2 are used to represent the structural components of the NEO information model. The ontology is a formal representation of domain knowledge as a logical theory consisting of classes, subclass relationships, properties, and assertions (such as class-disjointness). The core structure of the ontology consists of a hierarchy of entity-class concepts used to categorize phenomena in a universe of discourse. This taxonomic backbone of the NEO is derived from the hierarchy of entity types in the NAS. NEO properties are based on the presumed properties of real-world entities, and their values are constrained accordingly.

The use of OWL 2 allows the ontology to be expressed in a formal language that supports machine-processing. The specific variant of OWL 2 used is OWL-DL, which supports tractable reasoning by inference engines. Reasoners may be used both to evaluate the quality of the ontology as a domain model, and also to support the derivation of implicit knowledge about data instances by applying the domain theory to collected assertions.

The ontology also includes annotations that document the human-readable names and definitions of the modeled concepts. Annotation properties from OWL 2 and other recommendations – including the Simple Knowledge Organization System (SKOS) – are used to incorporate documentation of NAS-derived semantics within the ontology components.

The NEO ontology specification uses the OWL 2 Structural Specification in the following ways:

1. OWL Classes are used to represent NEO Entity and Datatype Classes.
2. OWL Object Property Expressions are used to represent NEO Entity Relationships and some NEO Entity Attributes.
3. OWL Data Property Expressions are used to represent the remaining NEO Entity Attributes.
4. OWL Class expressions are used to represent class unions and intersections; property restrictions; and cardinality restrictions (OWL 2 Structural Specification, Section 8).
5. OWL Class axioms are used to represent SubClassOf and DisjointClasses (OWL 2 Specification, Section 9.1).
6. OWL Object Property axioms are used to represent property Domain and Range declarations (OWL 2 Specification, Section 9.2 and 9.3).
7. OWL Datatypes are used (OWL 2 Specification, Sections 4 and 7) to represent NEO primitive datatypes.
8. Several OWL 2 Annotation properties (viz., `rdfs:label`, `rdfs:isDefinedBy`, `owl:versionInfo`) are used for NEO DocumentationProperty.

The NEO ontology specification also reuses annotation properties from other standards; these annotation properties are listed in Section 5.3.6.

5.3.3 Representing NEO Information Model Concepts in OWL

The standard ISO 19150-2, *Geographic information – Ontology – Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)*, defines a rule-based transformation from ISO 19100-series compliant UML application schemas to OWL 2 ontologies. The NEO Standard relies on correlations defined by those transformation rules in order to determine the OWL 2 representation of modeling elements in the NEO information model.

²¹ Linked Data Glossary, W3C Working Group Note 27 June 2013. (<http://www.w3.org/TR/ld-glossary/>)

²² Guidance from the W3C Data Activity acknowledges the importance of standards to provide semantics for shared data. All semantic resources are presented under the topic of “Vocabularies”, including ontologies and taxonomies as well as controlled terminologies.

²³ Accessible online at: <http://www.w3.org/TR/owl2-syntax/>.

The rules specified in ISO 19150-2 are the basis for determining the representation for most elements of the NEO information model in terms of constructs from OWL 2 and other Semantic Web standards, as listed in Table 10, Table 11, and Table 12. Where ISO 19150-2 does not address the OWL representation for an aspect of the ISO 19109 GFM model (especially, the disjointness of sibling subclasses indicated by 'uniqueInstance'), an OWL 2 encoding was developed (see Section 5.4.3.4).

5.3.4 Unique Identifiers in OWL: IRIs

OWL 2 ontologies and ontology elements are identified using Internationalized Resource Identifiers (IRIs). An IRI is a finite-length sequence of zero or more characters used for identifying an abstract or physical resource. A resource can be anything that has identity. IRIs may be used solely for identification of resources, or they may also be used to locate and access resources.

NOTE An Internationalized Resource Identifier is a sequence of characters from the Universal Character Set (Unicode/ISO 10646) that forms an identifier for a resource. IRIs complement an older format, Uniform Resource Identifiers (URIs), which allows the use of only a subset of the ASCII character set to construct identifiers. A standardized mapping from IRIs to URIs is defined in the IRI specification. When a resource identifier is used for resource retrieval, it may be necessary to determine the associated URI, because retrieval mechanisms may be defined only for URIs. Every URI is by definition an IRI.²⁴

An encoding of the NEO content is a Web resource identified by an IRI that is a URI. In addition, each component in the encoded NEO content – including its classes, properties, and individuals – is a resource and, as such, is identified by an IRI that is a URI.

The use of URIs for the NEO content and its components is consistent with requirements for the identification of resources on the internet. The intent of the World Wide Web is to enable sharing of locatable resources across a global community with both known and unanticipated users. Information-sharing is supported by use of a single global identification system that provides a common basis for unique identification of resources across the Web. Identification of Web resources by IRIs and URIs is a recommended practice of the World Wide Web Consortium (W3C). Benefits of using URIs to locate resources include linking, caching, bookmarking, and indexing by search engines. Key to the use of URIs is that each URI should identify a single distinct resource.²⁵

5.3.5 NEO Structural Elements in OWL

The structural elements of NEO include its entity classes, generalization relationships, subclass-disjointness axioms, characteristics of entities (including their attributes and relationships), and types of data values. The main Semantic Web construct used to represent each of these kinds of elements is identified below. Details of the encoding are discussed in Section 5.4.

- The NEO content comprises an `owl:Ontology` of NEO entities (with properties) and an `owl:Ontology` of NEO enumerations (specifying their allowed listed values).
- Entity classes in the NEO content are represented by instances of `owl:Class`.
- Class generalizations in the NEO content are represented by the property `rdfs:subClassOf`. The NEO information model represents only the type-generalization direction of the Inheritance Relation in the ISO 19109 GFM. The inverse type-specialization concept is logically implicit in the interpretation of the subclass property and may be logically inferred.
- Disjoint sibling subclasses are indicated in the ISO 19109 GFM by the Boolean property 'uniqueInstance'. This is represented in the encoded NEO content by OWL class axioms using `owl:AllDisjointClasses` to enumerate the classes that are disjoint. Disjoint classes have no individuals as members in common.
- Characteristics of entities in the NEO content are represented by instances of either `owl:DatatypeProperty` or `owl:ObjectProperty` (depending on the value type of the property).

²⁴ See: <http://tools.ietf.org/html/rfc3987#section-3.1>.

²⁵ Resources are broadly inclusive of Web pages, images, concepts, and even real-world objects. *Architecture of the World Wide Web, Volume One*. W3C Recommendation 15 December 2004. Ian Jacobs and Norman Walsh, Eds. Available online at: <http://www.w3.org/TR/webarch/#identification>.

- Datatypes in NEO include XSD datatypes and also specialized datatypes from the NAS, including enumerated types, measures, complex datatypes, and datatypes with metadata. Enumerated types are represented using `skos:Concept` and `skos:ConceptScheme`.

5.3.6 NEO Documentation Properties in OWL

The encoded NEO content includes information intended for human consumption as well as for machine reasoning. This documentation is represented using annotation properties for OWL 2 (listed in Section 5.3.2 above) and several other information standards. The annotation properties used for NEO documentation are presented below, grouped by standard. Their specific use with NEO components is explained in the presentation of the NEO encoding.

- **OWL**
 - `owl:versionInfo`
- **RDF Schema**
 - `rdfs:label`
 - `rdfs:isDefinedBy`
- **SKOS**
 - `skos:altLabel`
 - `skos:definition`
 - `skos:prefLabel`
- **ISO 19150-2**
 - `iso19150-2:associationName`
 - `iso19150-2:constraint`
- **Dublin Core Terminology**
 - `dct:isPartOf`²⁶
 - `dct:source`

The documentation properties are applied to the NEO as a whole and also to its components, as specified in the NEO information model presented in Section 5.2.2.

5.3.7 NEO Datatypes in OWL

NEO primitive datatypes are represented using OWL 2 datatypes. However, the datatypes available in OWL 2 are limited when compared to the spectrum of datatypes defined in the NEO information model (Sections 5.2.8.13 through 5.2.8.20).

From the *OWL 2 Structural Specification* (Section 5.2):

An IRI used to identify a datatype in an OWL 2 DL ontology *must*

- be *rdfs:Literal*, or
- identify a datatype in the OWL 2 datatype map (see Section 4), or
- not be in the reserved vocabulary of OWL 2 (see Section 2.4).

The conditions from the previous paragraph and the restrictions on datatypes in Section 11.2 require each datatype in an OWL 2 DL ontology to be *rdfs:Literal*, one of the datatypes from Section 4, or a datatype defined by means of a datatype definition (see Section 9.4).

The NEO information model requires complex datatypes in order to represent NAS datatypes.²⁷ Therefore, this NEO Standard defines and encodes additional datatypes, using OWL classes and properties as described in Section 5.4.4.

²⁶ This property is used only in the publication of REST API-accessible ontology resources (see Section 6.3.3).

²⁷ NAS – Part 1, Section 4.1, Figure 3.

5.4 NEO Content Encodings

5.4.1 Introduction

The NEO Standard specifies two technology-specific encodings of the NEO content that each accurately represents the information model in the NEO Standard:

- RDF/XML encoding – RDF/XML is the primary concrete exchange syntax for OWL 2, as specified in the W3C Recommendations. All OWL 2 tools are required to support the OWL 2 RDF/XML syntax (see Section 2.1 of the OWL 2 Conformance document²⁸).
- N-Triples – N-Triples is a line-based, plain-text format for encoding an RDF graph that may be used in information exchange without necessitating the complicated parsing required for RDF/XML. In N-Triples, each line consists of a sequence of three RDF terms representing, respectively, the Subject, Predicate, and Object of an RDF triple.²⁹

Some aspects of the NEO content encoding are the same for both technologies.

The following sections present the general aspects of the NEO content encodings first, followed by specific differences for the RDF/XML and N-Triples encodings. Both encodings use IRIs as described in Section 5.4.2 to uniquely identify ontology elements. Section 5.4.3 specifies the general approach for encoding the entity modeling concepts of the NEO information model in OWL 2. Section 5.4.4 describes the encoding of datatypes. Section 5.4.5 lists the differences between RDF/XML and N-Triples encodings of the NEO content.

The RDF/XML encoding employs the approach defined in ISO 19150-2:2015, as implemented and extended in the OGC Testbed-12 ShapeChange Engineering Report (OGC 16-020). The N-Triples encoding is derived from the RDF/XML encoding.

5.4.2 Namespace and Identifiers

5.4.2.1 Introduction

In the World Wide Web, resources must be uniquely identified by IRIs. Related resources may be grouped together into a “namespace” using a specified IRI structure for all resources in the namespace. The NEO content is a web resource that uses two namespaces, whose IRIs are specified in this Standard.

5.4.2.2 Namespaces

A namespace identifies a collection of resources by referencing them using identifiers (IRIs) that share a common initial prefix or “stem” (also referred to as a URI base). An RDF namespace is represented by a URI base used in all identifiers for a set of related resources. The namespace URI base is concatenated with a separator followed by a local name to create the complete IRI identifier for an RDF resource.³⁰

Resources from different namespaces may be referenced in the specification of a new resource. Modeling elements from multiple W3C Web resources are used for the representation of NEO content. Every modeling element from RDF, RDFS, OWL, and SKOS that is used in encodings of NEO content has a unique IRI that identifies that element in relation to its namespace. For example, the OWL concept Class (which has the IRI <http://www.w3.org/2002/07/owl#Class>) is in the OWL namespace (<http://www.w3.org/2002/07/owl#>).

NOTE The NEO Standard and NEO content in RDF/XML use prefix abbreviations for common namespaces as declared in the OWL Structural Specification (Section 2.4); e.g., 'owl' for the namespace identified by <http://www.w3.org/2002/07/owl#>. NEO content in N-Triples uses only fully specified namespace prefixes.

Components of the NEO content have a URI base that identifies them as belonging to one of the two NEO content namespaces. NEO entity classes and their properties are defined in one namespace, while NEO enumerations are specified in a separate namespace. The identifiers for these namespaces are specified in the next section.

²⁸ OWL 2 Web Ontology Language Conformance (Second Edition). W3C Recommendation. 11 December 2012. Michael Smith, *et al.*, eds. Published online at: <http://www.w3.org/TR/2012/REC-owl2-conformance-20121211/>.

²⁹ RDF 1.1. N-Triples. W3C Recommendation. 25 February 2014. David Beckett. Published online at: <http://www.w3.org/TR/n-triples/>.

³⁰ In the concatenation of a URI base with a local name, a separator (which may be either the hash (“#”) or the forward slash (“/”)) character is required between the two parts. The type of separator used depends upon the supported retrieval mechanism.

5.4.2.3 Identifiers

This NEO Standard assigns a unique URI base for each of the two namespaces in which NEO content is specified.

- URI base for NEO entity classes (with properties): <http://api.nsgreg.nga.mil/ontology/neo-ent>
- URI base for NEO enumerations: <http://api.nsgreg.nga.mil/ontology/neo-enum>

The former contains the entity concepts of the ontology (with their properties), while the latter contains the enumerations (with their listed values) defined for use with NEO properties.

Each entity class belonging to the NEO content has a unique identifier which embeds the first URI base, above. A NEO entity-class identifier is the combination of that URI base and (following a “#” separator) a unique terminal label for the concept (e.g., ‘Building’, ‘MountainPass’).

Each enumerated type belonging to the NEO content has a unique identifier which embeds the second URI base, above. A NEO enumerated-type identifier is the combination of that URI base and (following a “/” separator) a unique terminal label for the concept (e.g., ‘AerodromePhysicalConditionType’, ‘BuoyBuoyShapeType’).

The full patterns for NEO identifiers are specified in Section 5.4.2.5.

NEO IRIs are in the form of a Uniform Resource Locator (URL). A URL specifies the location of, and access to, a resource on the Internet. A URL specifies the protocol of the resource (e.g., ‘http’ or ‘ftp’), the domain name for the resource (e.g., ‘nsgreg.nga.mil’), and the relative location of the resource within that domain. If the site host is active, then accessing the specified resource results in retrieval of a representation (i.e., the content) of the resource; however, site persistence is not guaranteed.

5.4.2.4 Versioned and Non-versioned IRIs for NEO

The NEO Standard specifies both versioned and non-versioned IRIs for the NEO content.

The versioned IRI shall be used for authoritative identification of NEO content in information exchange and data sharing. The versioned IRI shall also be used for official specification of the NEO content baseline to be used in systems development and acquisition. Examples of the versioned IRI:

- Versioned IRI for ‘neo-ent’ (example): <http://api.nsgreg.nga.mil/ontology/neo-ent/1-3>
- Versioned IRI for ‘neo-enum’ (example): <http://api.nsgreg.nga.mil/ontology/neo-enum/1-3>

For convenience, the REST API component of the NSG Standards Registry supports the retrieval of the latest version of the NEO content when the non-versioned IRI is used.

- Non-versioned IRI for ‘neo-ent’: <http://api.nsgreg.nga.mil/ontology/neo-ent>
- Non-versioned IRI for ‘neo-enum’: <http://api.nsgreg.nga.mil/ontology/neo-enum>

Non-versioned IRIs (where supported) shall be used only as a shortcut that is redirected by the HTTP resolver to the latest versioned resource. The *version* pattern component is required in a resolved NEO IRI.

5.4.2.5 Form of IRIs for NEO Content

The IRIs for NEO content are constructed in accordance with the following pattern:

protocol “:” *domain* “/” *resource-type* “/” *resource* “/” *version* [“#” | “/”] *concept* [“/” *subconcept*]

Each component in the pattern is case-sensitive and determined as follows:

- *protocol* – always ‘http’
- *domain* – always ‘api.nsgreg.nga.mil’
- *resource-type* – always ‘ontology’
- *resource* – ‘neo-ent’ when the *concept* is a NEO entity class or property; ‘neo-enum’ when the *concept* is a NEO enumerated type or listed value
- *version* – designates the version of the resource (e.g., ‘1-1’, ‘1-2’)
- *concept* – designates a specific concept (e.g., ‘Aerial’, ‘AdministrativeBoundary.length’, ‘ApronAccessibilityStatusType’)

- *subconcept* (optional) – designates a listed value of an enumerated type (e.g., 'locked' is a subconcept of 'ApronAccessibilityStatusType').

In IRIs for NEO content, the separator between *version* and *concept* is assigned conditionally as follows:

- The separator “#” is used with each *concept* in the 'neo-ent' namespace (i.e., the *concept* is an entity class or property).
- The separator “/” is used with each *concept* in the 'neo-enum' namespace (i.e., the *concept* is an enumerated type or listed value).

The components described above are concatenated into a single string as specified by the pattern (above), to form the IRI that designates the ontology component. Examples for different types of ontology components:

http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Aerial	(The OWL class representing the entity class 'Aerial' in the NEO content, version 1-3)
http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Building	(The OWL class representing the entity class 'Building' in the NEO content, version 1.1)
http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#AdministrativeBoundary.length	(The OWL property representing the entity property 'AdministrativeBoundary.length' in the NEO content, version 1-3)
http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType	(The OWL class representing the enumeration 'ApronAccessibilityStatusType' in the NEO content, version 1-3)
http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked	(The OWL class representing the listed value of 'locked' for 'ApronAccessibilityStatusType' in the NEO content, version 1-3)

5.4.3 General NEO Encoding

5.4.3.1 Introduction

The NEO information model in Section 5.2 defined eight basic categories of information modeling concepts: Ontology, EntityClass, DisjointClasses, EntityProperty (including EntityAttribute and EntityRelationship), DataType, and DocumentationProperty. This section specifies the encodings for NEO modeling concepts using OWL 2 and other Semantic Web standards.

Abstract classes are not encoded. For example, the class EntityProperty (an abstract class) is not represented in the encoding; instead, its concrete subclasses are encoded. The class DocumentationProperty is an abstract class, which is not represented in the encoding. Instead, the specific documentation properties are assigned to the appropriate modeling concepts in Table 10, Table 11, and Table 12, and encoded with them.

The Ontology and each EntityClass concept have several types of information specified, including required and optional attribution. The IRI-valued attributes provide identification of the ontology components. The remaining attribution provides the means of further specifying an information modeling element by attributes and relationships.

Encoding elements for the NEO information model components are specified in the tables in 5.4. The table format used to document these encoding elements is as follows:

- The **Reference** column consists of a sequentially-assigned, non-normative identifier of the element (class or property) that is provided for cross-referencing purposes. It may vary from version-to-version of this document.
- The **NEO Modeling Concept** column specifies the class name or property name of the information modeling concept being encoded.
 - If the modeling concept is a class in the NEO information model, then the row is highlighted in light-grey.

- Properties are either attributes or relationships. Relationship names are prefixed by the italicized phrase “*Role name*”.
- The **NEO Encoding Element** column specifies the OWL 2 or other Semantic-Web standard construct that shall be used to represent the corresponding NEO Modeling Concept. NEO encoding elements are shown in the forms used by the (mandatory) RDF/XML encoding of the NEO content, which utilizes namespace abbreviations. (The optional N-Triples encoding requires the use of fully specified IRIs; see Section 5.4.5.3).
- The **Cardinality of Element** column indicates the number of occurrences of the element that are permitted by the information model. This column has values only for properties and is dark-grey filled on rows containing classes.
- The **Value Type** column indicates the modeling concept or specific datatype that is used to define the value(s) of the element. This column has values only for properties and is dark-grey filled on rows containing classes.
- The **Notes** column may contain comments, specifications of the actual value, or examples of values for the element.

5.4.3.2 Encoding of the Ontology

The Ontology modeling concept is used to represent the NEO ontology as a self-documenting resource. Each encoding of the NEO content consists of two Ontology entities ('neo-ent' and 'neo-enum') characterized by properties as specified in Section 5.2.2. The information model elements for the NEO content are encoded in OWL 2 and supporting W3C languages, as specified in Table 10.

Table 10 – Encoding Elements for the Ontology

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	Ontology	<code>owl:Ontology</code>			Used to represent each of the two namespaces for NEO content as a complete resource.
2	ontologyIRI	<code>rdf:about</code>	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-ent/1-3
3	versionIRI	<code>owl:versionIRI</code>	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-ent/1-3
4	versionInfo	<code>owl:versionInfo</code>	Exactly one	CharacterString	For example: "1-3"
5	label	<code>rdfs:label</code>	Exactly one	LocalizedContinuousString	The value is "NsgEnterpriseOntology".
6	name	<code>skos:prefLabel</code>	Exactly one	LocalizedCharacterString	The value is "NSG Enterprise Ontology".
7	alias	<code>skos:altLabel</code>	Zero or more	LocalizedCharacterString	For example: "NEO"
8	definitionNote	<code>skos:definition</code>	Exactly one	LocalizedCharacterString	The value is "Definition: The NSG Enterprise Ontology (NEO) Standard defines a logic-based specification in the W3C Web Ontology Language (OWL 2) of the domain model for Geospatial Intelligence (GEOINT) information shared in the U.S. National System for Geospatial Intelligence (NSG). Description: The NSG Enterprise Ontology contains a computer-interpretable representation of entity classes, relationships, datatypes, and constraints based on (and derived from) the NSG Enterprise Data model (i.e., NSG Application Schema (NAS)), which is implemented in two types of OWL 2 encodings: RDF/XML and N-Triples."
9	sourceIRI	<code>rdfs:isDefinedBy</code>	Exactly one	IRI	For the NEO Standard, the URL is: http://nsgreg.nga.mil/doc/view?i=2615
10	sourceTitle	<code>dct:source</code>	Exactly one	LocalizedCharacterString	The value is "NSG Enterprise Ontology (NEO) Standard".
11	<i>Role name:</i> dependency	<code>owl:imports</code>	Zero or more	Ontology	The imported ontology is represented by its IRI. For example: http://def.isotc211.org/iso19115/-/1/2014/MetadataInformation

5.4.3.3 Encoding of EntityClass

The EntityClass modeling concept is used to represent sets of individuals that share the same nature and specific properties. An EntityClass is characterized by the properties specified in Section 5.2.4, and selected documentation properties defined in 5.2.7. The information model elements for an EntityClass are encoded in OWL 2 and supporting W3C languages as specified in Table 11.

Table 11 – Encoding Elements for EntityClass

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	EntityClass	owl:Class			Used to represent entity types in NEO content
2	classIRI	rdf:about	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome
3	isAbstract	iso19150-2:isAbstract	If applicable, then exactly one.	Boolean	FALSE (By policy: FALSE, unless asserted as TRUE.)
4	label	rdfs:label	Exactly one	LocalizedContinuousString	For example: 'LandAerodrome'
5	name	skos:prefLabel	Exactly one	LocalizedCharacterString	For example: 'Land Aerodrome'
6	alias	skos:altLabel	Zero or more	LocalizedCharacterString	For example: 'Airport'
7	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	For example: (Land Aerodrome) "Definition: An aerodrome on land intended to be used either wholly or in part for the arrival, departure and surface movement of aircraft. Description: [None Specified]"
8	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	For example: http://nsgreg.nga.mil/as/view?i=100436 ³¹
9	constraint	iso19150-2:constraint	Zero or more	LocalizedCharacterString	The value is a structured string containing the name of the constraint rule, followed by a human-readable statement of the constraint.
9	<i>Role name:</i> generalization	rdfs:subClassOf	If applicable, then one or more.	EntityClass	For example: (the generalization of Land Aerodrome) http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Aerodrome

³¹ The EntityClass sourceIRI is a URL that incorporates the numerical Item Identifier value for the corresponding NAS Entity Type. The sourceIRI values for the other types of ontology components also use this URL pattern.

5.4.3.4 Encoding of DisjointClasses

The DisjointClasses modeling concept is used to represent collections of entity classes that are pairwise disjoint; that is, no individual shall belong at the same time to more than one of the member classes in a specific collection. This concept represents the constraint that sibling subclasses in the ontology are disjoint.

The encoding of NEO content uses `owl:AllDisjointClasses` to represent all DisjointClasses axioms declaring class disjointness.³² Each DisjointClasses collection is characterized by the property 'disjointMember', defined in Section 5.2.5, which identifies two or more entity classes as members of the disjoint-class collection. In the NEO content, these collections contain a set of sibling subclasses, which shall not share members.

Table 12 – Encoding Elements for DisjointClasses

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	DisjointClasses	<code>owl:AllDisjointClasses</code>			Used to represent disjointness of sibling subclasses
2	<i>Role name:</i> disjointMember	<code>owl:members</code>	If applicable, then two or more.	EntityClass	

Both encodings of NEO content represent DisjointClasses using the encoding elements above. There are some differences in the implementation-specific encodings, however, which are discussed in Section 5.4.5.

5.4.3.5 Encoding of EntityAttribute

The EntityAttribute modeling concept represents a characteristic that describes an entity in terms of a data value. EntityAttribute inherits attribution from its abstract superclass, EntityProperty (see Table 7), and also has specialized properties (Table 8). Applicable documentation properties (Section 5.2.7), including name and source information for the attribute, are also encoded.

There are two encodings for EntityAttribute, depending upon the datatype used as the attribute's property range.

1. For any EntityAttribute whose range is a PrimitiveDatatype, the EntityAttribute is encoded as an `owl:DatatypeProperty` with the range encoded using the appropriate OWL 2 datatype. (See Table 13 for this encoding.)
2. For any EntityAttribute that has a non-primitive DataType as its range, the EntityAttribute is encoded as an `owl:ObjectProperty` with the range encoded as described in Section 5.4.4. (See Table 14 for this encoding.)

The following diagram shows the alternative encoding patterns for EntityAttribute.

- EntityAttributeA has the primitive datatype Real as its property range (value type), while
- EntityAttributeB has the class RealIntervalMeta as its property range. RealIntervalMeta is a subtype of DatatypeWithMetadata, a complex datatype.

The first EntityAttribute is encoded using `owl:DatatypeProperty`, while the latter is encoded using `owl:ObjectProperty`. This alternative encoding is due to the difference in how NEO and OWL handle datatypes (Section 5.2.8.1). The range of the property `owl:DatatypeProperty` (as defined by the W3C Recommendation) is limited to OWL 2 datatypes. When NEO datatypes represented as classes are the range of an EntityAttribute, that EntityAttribute must be encoded as an `owl:ObjectProperty`.

³² The NEO Standard uses `owl:AllDisjointClasses` to represent all class axioms declaring class disjointness between two or more classes. The OWL 2 Functional Syntax maps OWL DisjointClasses to RDF graphs using `owl:disjointWith` for two classes, and `owl:AllDisjointClasses` for three or more classes. The OWL Primer illustrates use of `owl:AllDisjointClasses` for the 2-class case. RDF graphs containing `owl:AllDisjointClasses` constructs with two or more classes are mapped to DisjointClasses in the OWL 2 Functional syntax. (OWL 2 Mapping to RDF, Section 3.2.5)

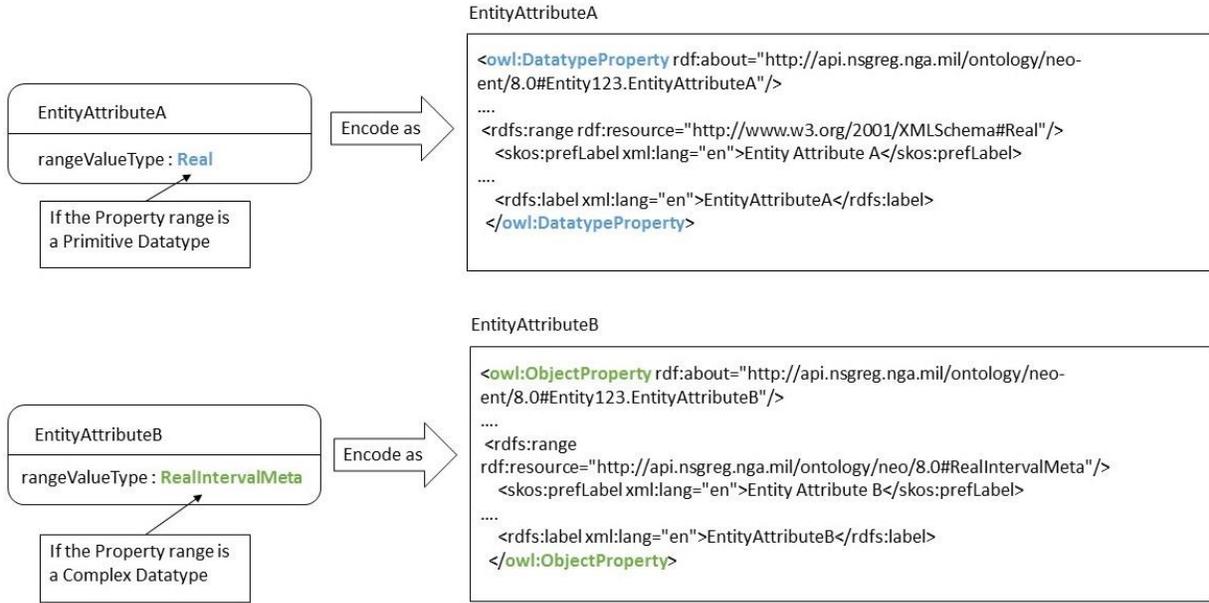


Figure 4 – Range-based Alternative Encodings for EntityAttribute

Table 13, below, presents the OWL 2 encoding for an EntityAttribute that has a primitive datatype as its property range.

Table 13 – Encoding Elements for EntityAttribute (with PrimitiveDatatype Range)

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	EntityAttribute	owl:DatatypeProperty			EntityAttribute with a primitive datatype as range.
2	propertyIRI	rdf:about	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#ElectronicRecordsManagement.vitalRecordsIndicator
3	label	rdfs:label	Exactly one	LocalizedContinuousString	For example: "ElectronicRecordsManagement.vitalRecordsIndicator"
4	name	skos:prefLabel	Exactly one	LocalizedCharacterString	For example: "Electronic Records Management Information : Vital Record Indicator"
5	alias	skos:altLabel	Zero or more	LocalizedCharacterString	For example: "essential record"
6	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	For example (for ElectronicRecordsManagement.vitalRecordsIndicator): "Definition: An indication that a managed record is considered essential to continuity of operation during and after emergencies or disaster conditions. Description: Also known as an Essential Record (as specified in the U.S. Federal Continuity Directive 1 (FCD 1) 2012)."
7	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	For example: http://nsgreg.nga.mil/as/view?i=194725
8	rangeValueType	rdfs:range	Exactly one	PrimitiveDatatype	For example (for ElectronicRecordsManagement.vitalRecordsIndicator): http://www.w3.org/2001/XMLSchema#boolean
9	<i>Role name:</i> characterizedEntity	rdfs:domain	Zero or more	EntityClass or DataType	A class representing a non-primitive DataType may have properties. The EntityClass is referenced by its IRI. For example (for the domain of ElectronicRecordsManagement.vitalRecordsIndicator): http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#ElectronicRecordsManagement

Table 14, below, presents the OWL encoding for an EntityAttribute that has a non-primitive datatype as its property range.

Table 14 – Encoding Elements for EntityAttribute (with non-PrimitiveDatatype Range)

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	EntityAttribute	<code>owl:ObjectProperty</code>			EntityAttribute with a non-primitive datatype as range.
2	propertyIRI	<code>rdf:about</code>	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Building.featureFunction
3	label	<code>rdfs:label</code>	Exactly one	LocalizedContinuousString	For example: "Building.featureFunction"
4	name	<code>skos:prefLabel</code>	Exactly one	LocalizedCharacterString	For example: "Building : Feature Function"
5	alias	<code>skos:altLabel</code>	Zero or more	LocalizedCharacterString	For example: "purpose"
6	definitionNote	<code>skos:definition</code>	Exactly one	LocalizedCharacterString	For example (for Building.featureFunction): "Definition: A purpose of, or intended role served by, a feature. Description: [None Specified]"
7	sourceIRI	<code>rdfs:isDefinedBy</code>	Exactly one	IRI	For example: http://nsgreg.nga.mil/as/view?i=101855
8	rangeValueType	<code>rdfs:range</code>	Exactly one	<i>DataType {Abstract}</i> [excluding PrimitiveDatatype]	The datatype is referenced by its IRI. For example (for Building.featureFunction): http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#BuildingFeatureFunctionCodeMeta
9	<i>Role name:</i> characterizedEntity	<code>rdfs:domain</code>	Zero or more	EntityClass	The EntityClass is referenced by its IRI. For example (for Building.featureFunction): http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Building

5.4.3.6 Encoding of EntityRelationship

The EntityRelationship modeling concept represents a characteristic that describes an entity in terms of its association with another entity. EntityRelationship inherits the properties from its generalization in the NEO information model (see Table 7), and also has the specialized properties defined in Table 9. Applicable documentation properties (Section 5.2.7), including name and source information for the relationship, are also encoded.

EntityRelationships are encoded using `owl:ObjectProperty`. It should be noted that in the NEO encoding, an `owl:ObjectProperty` can represent either an EntityAttribute or an EntityRelationship. In the former case, the range of the `owl:ObjectProperty` will be a NEO non-primitive DataType. In the latter case, the range of the `owl:ObjectProperty` will be a NEO EntityClass that is not a DataType.

The property `owl:inverseOf` is used only with encodings of EntityRelationship. Each EntityRelationship that models a relationship between an individual and a relationship with properties shall have an `owl:inverseOf` property.

Table 15 – Encoding Elements for EntityRelationship

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	EntityRelationship	owl:ObjectProperty			An EntityProperty relating individuals that are instances of classes.
2	propertyIRI	rdf:about	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo/1-3#LandAerodrome.railway
3	label	rdfs:label	Exactly one	LocalizedContinuousString	For example: "LandAerodrome.railway"
4	name	skos:prefLabel	Exactly one	LocalizedCharacterString	For example: "Land Aerodrome-associated Railway"
5	alias	skos:altLabel	Zero or more	LocalizedCharacterString	For example: "airport railway"
6	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	For example (for LandAerodrome.railway): "Definition: A railway that is associated with this land aerodrome (for example: passes through or is located within its perimeter). Description: [None Specified]"
7	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	For example: http://nsgreg.nga.mil/as/view?i=126027 ³³
8	<i>Role name:</i> characterizedEntity	rdfs:domain	Zero or more	EntityClass	The EntityClass is referenced by its IRI. For example (for LandAerodrome.railway): http://api.nsgreg.nga.mil/ontology/neo/1-3#LandAerodrome
9	<i>Role name:</i> hasRangeEntityClass	rdfs:range	Zero or more	EntityClass	The EntityClass is referenced by its IRI. For example (for LandAerodrome.railway): http://api.nsgreg.nga.mil/ontology/neo/1-3#Railway
10	<i>Role name:</i> inverseOf	owl:inverseOf	Zero or one	EntityRelationship	For example, "http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Runway.intersection" is the inverse of "http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#RunwayIntersection.runway".

As described in Section 5.2.6, NEO relationships that have properties (for example, the representation of a bilateral alliance relationship that was in place from 2008-2014) are modeled using a specific complex pattern.³⁴ The pattern for encoding this kind of relationship represents the relationship using two OWL Classes; there is a class for each direction of the relationship, and they are related as inverses. Each relationship Class has roles with the two entities that it relates. This pattern is explained and illustrated in the ShapeChange Engineering Report, Section 8.2.5.

³³ The EntityClass IRI incorporates the numerical Item Identifier value for the corresponding NAS Entity Type.

³⁴ In UML, this concept is represented by a single modeling element, a UML AssociationClass.

5.4.4 General Encoding of Datatypes

5.4.4.1 Introduction

The datatypes included in the NEO information model fall under four general classes, which are explained in corresponding following sections:

1. Primitive Datatypes
2. Measure Datatypes
3. Enumerated Types
4. Complex Data Types.³⁵

As noted in the discussion of OWL (Section 5.3.7), OWL supports selected datatypes using `rdf:PlainLiteral` and XML Schema. The NEO content has a larger set of datatypes, as specified in Section 5.2.8.

The non-primitive NEO datatypes are encoded using `owl:Class` with the appropriate properties. The figure below depicts the upper level of NEO datatype classes.

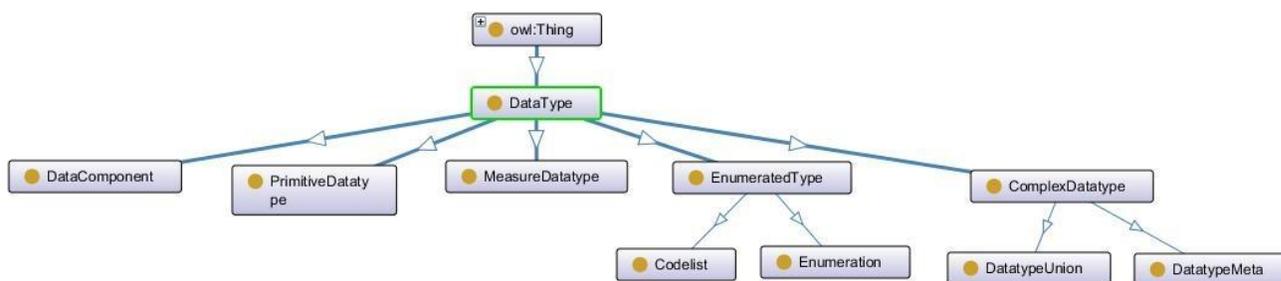


Figure 5 – NEO Datatype Hierarchy (Upper-level)³⁶

5.4.4.2 PrimitiveDatatype

The primitive datatypes included in the NEO information model are encoded as specified below. The abstract superclass Number, which serves as the generalization of the Real and Integer datatypes, is not encoded.

- The datatype `CharacterString` is represented using the `rdf:PlainLiteral` datatype provided in OWL 2 for the representation of strings optionally with an identified natural language, as specified in the OWL 2 Structural Specification and Functional-Style Syntax: Section 4.3. The `rdf:PlainLiteral` value may be either a character string (`xsd:String`), or an ordered pair consisting of a character string (`xsd:String`) and a lower-case language tag (e.g., “en” for English).³⁷
- The datatype `LocalizedCharacterString` is represented using the `rdf:PlainLiteral` datatype provided in OWL 2 for the representation of strings with the constraint that a language tag identifying the natural language of the content of the string is required.
 - The values of the datatype `IANALanguageSubtag`, used to identify the language of character strings, are represented using the two-character, lowercase language abbreviations specified in BCP 47.
- The datatype `LocalizedContinuousString` is represented using the `rdf:PlainLiteral` datatype provided in OWL 2 for the representation of strings, with the constraint that the string must not contain spaces (unless encoded by ‘%20’) and with the constraint that a language tag identifying the natural language of the content of the string is required.

³⁵ The NEO content encodings include all of the allowed specializations of the datatypes defined in the NEO information model.

³⁶ Image generated with the OWLViz plug-in to Protégé. Note that the graphical conventions in OWLViz differ from those of UML notation (specifically, the subclass arrows in OWLViz point towards the subclass, while they point towards the superclass in UML).

³⁷ W3C. *rdf:PlainLiteral: A Datatype for RDF Plain Literals*. 11 December 2012. Available online at: http://www.w3.org/TR/2012/REC-rdf-plain-literal-20121211/#Definition_of_the_rdf:PlainLiteral_Datatype.

- The datatype Boolean is represented using the XML datatype for OWL 2, `xsd:boolean`, as specified in the OWL 2 Structural Specification and Functional-Style Syntax.
- The datatype IRI is represented using the XML datatype `xsd:anyURI`, as specified in the OWL 2 Structural Specification and Functional-Style Syntax.
- The datatype DateTime is represented using the XML datatype `xsd:dateTime` as specified in the OWL 2 Structural Specification and Functional-Style Syntax.
- The datatype Real is represented using the XML datatype `owl:real`, as specified in the OWL 2 Structural Specification and Functional-Style Syntax.
- The datatype Decimal is represented using the XML datatype `xsd:decimal`, as specified in the OWL 2 Structural Specification.
- The datatype Integer is represented using the XML datatype `xsd:integer`, as specified in the OWL 2 Structural Specification and Functional-Style Syntax.
- The datatype NonNegativeInteger is represented using the XML datatype `xsd:nonNegativeInteger`, as specified in the OWL 2 Structural Specification and Functional-Style Syntax.

Table 16 – Encoding Elements for PrimitiveDatatype

Ref #	NEO Modeling Concept	NEO Encoding Element	Notes
1	<i>PrimitiveDatatype</i> { <i>Abstract</i> }	N/A	Abstract generalization of primitive datatypes used in the NEO information model.
2	CharacterString	rdf:PlainLiteral	W3C <i>rdf:PlainLiteral</i> (11 December 2012)
3	LocalizedCharacterString	rdf:PlainLiteral	Adds Language tag (from IANA LanguageSubtag).
4	LocalizedContinuousString	rdf:PlainLiteral	Adds Language tag (from IANA LanguageSubtag), and requires that the string does not contain white space (unless encoded using '%20').
5	Boolean	xsd:boolean	XSD Datatypes, 3.3.2
6	IRI	xsd:anyURI	OWL Functional Syntax, 4.6; XSD Datatypes, 3.3.17
7	DateTime	xsd:dateTime	XSD Datatypes, 3.3.7
8	Real	owl:real	OWL Functional Syntax, 4.1
9	Decimal	xsd:decimal	XSD Datatypes, 3.2.3
10	Integer	xsd:integer	XSD Datatypes, 3.3.14
11	NonNegativeInteger	xsd:nonNegativeInteger	XSD Datatypes, 3.3.20

5.4.4.3 MeasureDatatype

The datatype MeasureDatatype is used to represent a numeric amount (Number) expressed with a unit or scale or using a scalar reference system (Unit of Measure). MeasureDatatypes have both a measureValue and a measureUnit.

- Value: The abstract datatype Number is the generalization for the concrete datatypes Real, Decimal, Integer, and (a subclass of Integer) NonNegativeInteger. The abstract superclass Number is not encoded. Each of the concrete primitive datatypes is encoded as specified in Section 5.4.4.2.
- Unit of Measure: The datatype UnitOfMeasure used in the NEO content specifies units defined in ISO 80000 (multi-part) and encoded as IRI references to entries in the NSGREG Physical Quantities Register. For example: <http://api.nsgreg.nga.mil/physical-quantity/length/metre>.

MeasureDatatype-valued properties may be included within a ComplexDatatype (see Section 5.4.4.5) that enables measures to be collected together with an accuracy evaluation or other additional information (for example, whether a measurement of an interval represents an open or a closed interval).

Table 17 – Encoding Elements for MeasureDatatype

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	MeasureDatatype	owl:Class			All measure datatypes belong to one of the two concrete subclasses: iso19103_uom:Measure or iso19103_uom:DirectedMeasure
2	classIRI	rdf:about	Exactly one	IRI	For example: http://def.isotc211.org/iso19103/2015/MeasureTypes#Length
3	label	rdfs:label	Exactly one	LocalizedContinuousString	
4	name	skos:prefLabel	Exactly one	LocalizedCharacterString	
5	alias	skos:altLabel	Zero or more	LocalizedCharacterString	
6	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	
7	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	
8	measureValue	iso19103_uom:Measure. measureValue or iso19103_uom:Directed Measure.measureValue	Exactly one	Real (Measure) or Vector (DirectedMeasure)	
9	measureUnit	iso19103_uom:Measure. measureUnit or iso19103_uom:Directed Measure.measureUnit	Exactly one	IRI	The IRI identifies a unit of measure registered in the Physical Quantities Register of the NSG Standards Registry.

5.4.4.4 EnumeratedType

The datatype EnumeratedType represents sets of domain values (ListedValues) which are the allowed data values of an EntityAttribute. This datatype requires a complex encoding pattern. Each EnumeratedType is implemented as an OWL 2 class (owl:Class) that is a subclass of SKOS Concept, while an associated EnumeratedTypeScheme is implemented as a SKOS concept scheme (skos:ConceptScheme). Each ListedValue is represented as an individual SKOS concept (skos:Concept). The OWL class provides the typing (classification) for the listed values and serves as the range of an EntityAttribute. The SKOS concept scheme collects the listed values and enables them to be related by generalization relationships (skos:broader).

An enumerated type is either an Enumeration or a Codelist. The distinction is encoded using a Boolean property (neox:valuesComplete) on the associated EnumeratedTypeScheme (skos:ConceptScheme). The Boolean value TRUE indicates an Enumeration (i.e., a closed, non-extensible set of listed values), while the Boolean value FALSE indicates a Codelist (i.e., an extensible set of listed values).

The general encoding pattern for EnumeratedType, EnumeratedTypeScheme, and ListedValue is summarized as follows:

- An EnumeratedType is encoded as an owl:Class that may be used as the value type (rdfs:range) of an EntityAttribute.

- An EnumeratedTypeScheme is encoded as a `skos:ConceptScheme`. The concept scheme supports generalizations between listed values.
- The EnumeratedTypeScheme (`skos:ConceptScheme`) is related (unidirectionally) to the corresponding class (`owl:Class`) representation of the EnumeratedType, using `dct:isFormatOf` to encode the `schemeOf` relationship from the information model.
- Each ListedValue is encoded as a `skos:Concept` that is an instance of the `owl:Class` that encodes the EnumeratedType.
- A ListedValue may be related to another ListedValue that has a more general (*i.e.*, broader) meaning, by using `skos:broader`.
- The `hasMember` association role between an EnumeratedType and a ListedValue is represented by its inverse (“has member”) in SKOS. This relationship is encoded using `skos:inScheme` and is used to link each ListedValue to its EnumeratedTypeScheme (`skos:ConceptScheme`).
- If a ListedValue in an EnumeratedTypeScheme has no broader ListedValue, then the relationship `topMemberOf` (encoded by `skos:topConceptOf`) is also used to relate that ListedValue (`skos:Concept`) to its EnumeratedTypeScheme (`skos:ConceptScheme`).
- The Boolean-valued attribute `valuesComplete` (encoded by `neox:valuesComplete`) is used to distinguish an Enumeration from a Codelist. The distinction between an Enumeration and a Codelist is represented in the encoding by the value of the Boolean property (`neox:valuesComplete`) on the EnumeratedTypeScheme (`skos:ConceptScheme`). The value `TRUE` indicates an Enumeration, while the value `FALSE` indicates a Codelist.

EnumeratedTypes for NEO are specified in either the NEO enumeration namespace or in external namespaces. NEO Enumerations (and their ListedValues) are encoded in the ‘neo-enum’ namespace (for example <http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/AerodromeFacilityOperationalStatusType>). Codelists (and their ListedValues) that may be used as the value type (*i.e.*, range) for a NEO EntityAttribute shall be encoded according to the pattern specified in this section. These include codelists in the Information Resources (IR) Registry of the NSG Standards Registry. External codelists and listed values are referenced by IRIs from external namespaces; for example: <http://api.nsgreg.nga.mil/codelist/BuildingFeatureFunction/accommodation> and http://def.isotc211.org/iso19115-1/2014/CitationAndResponsiblePartyInformation#CI_Contact.

Table 18 – Encoding Elements for EnumeratedType

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	EnumeratedType	<code>owl:Class</code>			The class representing the EnumeratedType is also declared to be a subclass of <code>skos:Concept</code> .
2	<code>classIRI</code>	<code>rdf:about</code>	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-enum/base17May/AerodromePhysicalConditionType
3	<code>label</code>	<code>rdfs:label</code>	Exactly one	LocalizedContinuousString	
4	<code>name</code>	<code>skos:prefLabel</code>	Exactly one	LocalizedCharacterString	
5	<code>alias</code>	<code>skos:altLabel</code>	Zero or more	LocalizedCharacterString	
6	<code>definitionNote</code>	<code>skos:definition</code>	Exactly one	LocalizedCharacterString	
7	<code>sourceIRI</code>	<code>rdfs:isDefinedBy</code>	Exactly one	IRI	

An EnumeratedTypeScheme, encoded as specified below, is associated to each EnumeratedType.

Table 19 – Encoding Elements for EnumeratedTypeScheme

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	EnumeratedTypeScheme	skos:ConceptScheme			The EnumeratedType as a skos:ConceptScheme.
2	classIRI	rdf:about	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-enum/base17May/AerodromePhysicalConditionType_ConceptScheme
3	valuesComplete	neox:valuesComplete	Exactly one	Boolean	TRUE for Enumeration; FALSE for Codelist If unspecified, the default is FALSE (<i>i.e.</i> , open).
4	label	rdfs:label	Exactly one	LocalizedContinuousString	
5	name	skos:prefLabel	Exactly one	LocalizedCharacterString	
6	alias	skos:altLabel	Zero or more	LocalizedCharacterString	
7	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	
8	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	Same as the source for the associated EnumeratedType (owl:Class).
9	<i>Role name:</i> schemeOf	dct:isFormatOf	Exactly one	EnumeratedType	
10	<i>Role name:</i> hasTopMember	skos:hasTopConcept	Two or more	ListedValue	
11	<i>Role name:</i> hasMember	N/A	Two or more	ListedValue	SKOS has no encoding for this concept, which is represented by the inverse, skos:inScheme.

Allowed data values (ListedValue) are encoded using skos:Concept, as specified in the table below. Each ListedValue will be a direct instance of an EnumeratedType which is encoded by an owl:Class that is a subclass of skos:Concept. Each ListedValue will also be a member of the EnumeratedTypeScheme corresponding to that EnumeratedType.

Table 20 – Encoding Elements for ListedValue Datatype

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	ListedValue	skos:Concept			The ListedValue will be a direct instance of an owl:Class that is a subclass of skos:Concept.
2	classIRI	rdf:about	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-enum/base17May/AerodromePhysicalConditionType/amaged
3	label	rdfs:label	Exactly one	LocalizedContinuousString	
4	name	skos:prefLabel	Exactly one	LocalizedCharacterString	
5	alias	skos:altLabel	Zero or more	LocalizedCharacterString	
6	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	
7	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	
8	<i>Role name</i> memberOf	skos:inScheme	Exactly one	EnumeratedTypeScheme	
9	<i>Role name</i> topMemberOf	skos:topConceptOf	Exactly one	EnumeratedTypeScheme	
10	<i>Role name</i> broaderValue	skos:broader	Zero or one	ListedValue	For example: the listed value http://api.nsgreg.nga.mil/codelist/BuildingFeatureFunction/longTermAccommodation has the broader listed value http://api.nsgreg.nga.mil/codelist/BuildingFeatureFunction/accommodation

5.4.4.5 Complex Datatypes

A complex datatype is encoded in an identical way to an EntityClass (Section 5.4.3.3) with optional EntityProperty (Section 5.2.6).

A complex datatype has multiple properties, at least one of which provides a principal data value while others may contain additional contextual data (including metadata) about the principal data value. These multiple related properties together characterize an entity. For example, Elevation with Datum and Accuracy³⁸ is a complex datatype composed of a Real, a Vertical Datum, and an Absolute Vertical Accuracy; the first property specifies a real value for the elevation, while the second property specifies a vertical reference datum, and the third property specifies the accuracy of the elevation value.

The tabular encoding specifications in this section cover two subtypes of ComplexDatatype that represent common encoding patterns for complex datatypes used in the NEO content. First, the datatype DatatypeUnion represents a complex datatype consisting of a set of properties which are alternatives. Only one of the constituent properties is evaluated for any data instance. A common pattern of DatatypeUnion in the NEO content contains alternative properties that either provide the principal value(s) for a domain attribute or else a reason that the principal data value is absent.

³⁸ NAS Elevation with Datum and Accuracy (<http://nsgreg.nga.mil/as/view?i=100934>).

Table 21 – Encoding Elements for DatatypeUnion

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	DatatypeUnion	owl:Class			
2	classIRI	rdf:about	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-ent/base17May#MaritimeBottomCharacterSedimentColourCodeReason
3	label	rdfs:label	Exactly one	LocalizedContinuousString	
4	name	skos:prefLabel	Exactly one	LocalizedCharacterString	
5	alias	skos:altLabel	Zero or more	LocalizedCharacterString	
6	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	
7	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	
8	<i>Role name</i> value or values	XXX.value or XXX.values	Exactly one	[see Notes]	The specific datatype (“XXX”) for the value(s) property depends upon the particular DatatypeUnion. For example: http://api.nsgreg.nga.mil/ontology/neo-ent/base17May#MaritimeBottomCharacterSedimentColourCodeReason.value . The Value Type for value(s) also depends on the particular DatatypeUnion; for example: http://api.nsgreg.nga.mil/codelist/MaritimeBottomSedimentColour .
9	<i>Role name</i> reason	XXX.reason	Exactly one	VoidValueReason or VoidNumericValueReason	The specific datatype (“XXX”) for the reason property depends upon the particular DatatypeUnion. For example: http://api.nsgreg.nga.mil/ontology/neo-ent/base17May#MaritimeBottomCharacterSedimentColourCodeReason.reason . The reason Value Type is from a standardized codelist (either http://api.nsgreg.nga.mil/codelist/VoidValueReason or http://api.nsgreg.nga.mil/codelist/VoidNumericValueReason).

Second, the datatype DatatypeMeta represents a datatype having at least one property that provides a principal data value, accompanied (optionally) by properties providing metadata including, for example, restrictions on use, temporal extent, or provenance of the principal data value

Table 22 – Encoding Elements for DatatypeMeta

Ref #	NEO Modeling Concept	NEO Encoding Element	Cardinality of Element	Value Type	Notes
1	DatatypeMeta	owl:Class			
2	classIRI	rdf:about	Exactly one	IRI	For example: http://api.nsgreg.nga.mil/ontology/neo-ent/base17May#BuildingFeatureFunctionCodeMeta
3	label	rdfs:label	Exactly one	LocalizedContinuousString	
4	name	skos:prefLabel	Exactly one	LocalizedCharacterString	
5	alias	skos:altLabel	Zero or more	LocalizedCharacterString	
6	definitionNote	skos:definition	Exactly one	LocalizedCharacterString	
7	sourceIRI	rdfs:isDefinedBy	Exactly one	IRI	
8	<i>Role name</i> [see Notes]	[see Notes]	Zero or more	[see Notes]	Context-specific: Zero or more instances of EntityProperty, depending upon the specific subclass of DatatypeMeta.
9	<i>Role name</i> DatatypeMeta.metadata	DatatypeMeta.metadata	Zero or one	PropertyMetadata	
10	<i>Role name</i> DatatypeMeta.propertyValApplicableTime	DatatypeMeta.propertyValApplicableTime	Zero or one	TimeIntervalInfo	
11	<i>Role name</i> DatatypeMeta.resourceConstraints	DatatypeMeta.resourceConstraints	Zero or one	ResourceConstraints	
12	<i>Role name</i> DatatypeMeta.legalConstraints	DatatypeMeta.legalConstraints	Zero or one	LegalConstraints	

There are other kinds of complex datatypes in the NEO content; however, these two are the most common. Annex D (“Inspecting NEO Content”) presents an example of a DatatypeMeta complex datatype.

5.4.5 Technology-specific NEO Encodings

5.4.5.1 Introduction

The NEO Standard defines two technology-specific encodings of the NEO content, conformant with the information model of the NEO Standard:

- RDF/XML encoding – RDF/XML is the primary concrete exchange syntax for OWL 2. All OWL 2 tools are required to support the OWL 2 RDF/XML syntax (see Section 2.1 of the OWL 2 Conformance document³⁹). NEO conformance requires support for the RDF/XML encoding of NEO.
- N-Triples – A line-based, plain-text format for encoding an RDF graph.⁴⁰

Each encoding of NEO content provides a machine-interpretable OWL 2 representation of the entity-class generalization hierarchy for use in storing and exchange of geospatial information in Semantic Web applications. The NEO content in RDF/XML may be used by Semantic Web tools to enhance search or retrieval of instance data.

Data instance files in N-Triples encoding may be linked to related content in the N-Triples NEO encoding to provide semantics to that data when exchanging it among information systems or making it available in Linked Data stores.

The publication of the NEO content encodings and how to obtain them is described in Section 6.3.

5.4.5.2 RDF/XML Encoding

The NEO Standard specifies a technology-specific encoding for the NEO information model using the mandatory RDF/XML encoding of OWL 2.

- In the RDF/XML encoding, character strings in the ‘definitionNote’ (`skos:definition`) are encoded using the XML CDATA wrapper.⁴¹
- In the RDF/XML encoding, where language tags are required or permitted, they shall be provided as the value of an RDF/XML annotation element (`xml:lang`) for the string-valued property, in order to indicate that the content is in English (language code “en”).
- In OWL encoded in RDF/XML, assertions about disjoint sibling subclasses are expressed using the class expression `owl:AllDisjointClasses` with a list of the disjoint classes. The encoding specified in Section 5.4.3.4 is used for all collections of disjoint classes, whether there are two or more disjoint classes.

³⁹ OWL 2 Web Ontology Language Conformance (Second Edition). W3C Recommendation. 11 December 2012. Michael Smith, *et al.*, eds. Published online at: <http://www.w3.org/TR/2012/REC-owl2-conformance-20121211/>.

⁴⁰ RDF 1.1. N-Triples. W3C Recommendation. 25 February 2014. David Beckett. Published online at: <http://www.w3.org/TR/n-triples/>.

⁴¹ In XML the CDATA wrapper is used to indicate to parsers that the enclosed content should not be further interpreted; this allows applications to use characters in data exchange that would otherwise be misinterpreted as element or entity markup.

```

<owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Aerodrome">
  <rdfs:subClassOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#FeatureEntity"/>
  <iso19150-2:isAbstract rdf:datatype=http://www.w3.org/2001/XMLSchema#boolean>true</iso19150-2:isAbstract>
  <skos:prefLabel xml:lang="en">Aerodrome</skos:prefLabel>
  <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=103150"/>
  <skos:definition xml:lang="en">![CDATA[Definition: A defined area on land or water (including any buildings, installations and equipment) intended to be used either wholly or in part for the arrival, departure and surface movement of aircraft. Description: [None Specified]]]</skos:definition>
  <rdfs:label xml:lang="en">Aerodrome</rdfs:label>
</owl:Class>

<owl:AllDisjointClasses>
  <owl:members rdf:parseType="Collection">
    <owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Heliport"/>
    <owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome"/>
    <owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#WaterAerodrome"/>
  </owl:members>
</owl:AllDisjointClasses>

```

Figure 6 – OWL2 RDF/XML Encoding: Entity Class Aerodrome and its Disjoint Subclasses

```

<owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Helicopter">
  <rdfs:subClassOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Aerodrome"/>
  <skos:prefLabel xml:lang="en">Helicopter</skos:prefLabel>
  <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=100442"/>
  <skos:definition xml:lang="en">![CDATA[Definition: An aerodrome intended to be used for the arrival, landing, takeoff or departure
of vertical takeoff and landing aircraft/helicopters. Description: [None Specified]]]</skos:definition>
  <rdfs:label xml:lang="en">Helicopter</rdfs:label>
</owl:Class>

<owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Aerodrome"/>
  </rdfs:subClassOf>
  <skos:altLabel xml:lang="en">Airfield</skos:altLabel>
  <skos:altLabel xml:lang="en">Airport</skos:altLabel>
  <skos:prefLabel xml:lang="en">Land Aerodrome</skos:prefLabel>
  <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=100436"/>
  <skos:definition xml:lang="en">![CDATA[Definition: An aerodrome on land intended to be used either wholly or in part for the
arrival, departure and surface movement of aircraft. Description: [None Specified]]]</skos:definition>
  <rdfs:label xml:lang="en">LandAerodrome</rdfs:label>
</owl:Class>

<owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#WaterAerodrome">
  <rdfs:subClassOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Aerodrome"/>
  <skos:altLabel xml:lang="en">Sea Plane Base</skos:altLabel>
  <skos:prefLabel xml:lang="en">Water Aerodrome</skos:prefLabel>
  <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=100452"/>
  <skos:definition xml:lang="en">![CDATA[Definition: An aerodrome intended to be used either wholly or in part for the arrival,
departure and surface movement of aircraft on water. Description: [None Specified]]]</skos:definition>
  <rdfs:label xml:lang="en">WaterAerodrome</rdfs:label>
</owl:Class>

```

Figure 7 – OWL 2 RDF/XML Encoding: Subclasses of Aerodrome

5.4.5.3 N-Triples Encoding

The NEO Standard specifies a technology-specific encoding for the NEO information model using N-Triples. The W3C Recommendation RDF 1.1 N-Triples specifies a line-based, plain text format for encoding an RDF graph with each triple presented on a separate line followed by a period. N-Triples files do not contain special parsing instructions.

The encoding of NEO content in N-Triples closely follows the general encoding for the NEO information model, with the following technology-specific encodings applied:

- Namespace abbreviations are not used in the N-Triples encoding; instead, fully-specified IRIs are used. For example: `http://www.w3.org/2002/07/owl#AllDisjointClasses`, rather than `owl:AllDisjointClasses`. IRIs are text only; they are not hyperlinked.
- In the N-Triples encoding, the first component (subject) of the triple corresponds to the value of the `rdf:about` in the RDF/XML encoding. (`rdf:about` is not used.)
- In the N-Triples encoding, the XML CDATA wrapper is not used (e.g., with the NEO modeling element 'definitionNote').
- In the NEO N-Triples encoding, where language tags are required or permitted, they shall be appended to the character string by using the '@' symbol, in order to indicate that their content is in English (language code "en").
- In the N-Triples encoding, assertions about disjoint sibling subclasses are expressed using the class expression `http://www.w3.org/2002/07/owl#AllDisjointClasses` and a list of the disjoint classes. This encoding is used for all collections of disjoint classes, whether there are two or more disjoint classes. See the example in Figure 9, below
- The encoding of `owl:AllDisjointClasses` in OWL N-Triples results in the use of blank nodes to represent: (1) an instance of `owl:AllDisjointClasses`, and (2) the declarations of each member of the list of the disjoint classes. A Skolemized IRI is substituted as the identifier for the list of disjoint classes.

A blank node is a node in an RDF graph that has no IRI identifier. Blank nodes have labels beginning with an underscore character followed by a colon ("_:"). These are not IRIs and cannot be referenced outside of the local graph.

For some applications, it is valuable to assign a unique identifier to the `owl:AllDisjointClasses` construct. This is accomplished through a process termed "Skolemization". In order to reference a blank node, the label for that node is replaced with a new, skolemized, globally unique IRI corresponding to the blank node. In the NEO content, Skolemized IRIs are character strings beginning with the URI base 'http://api.nsgreg.nga.mil/.well-known/genid/', followed by a Universally Unique Identifier (UUID). Other blank nodes used to represent the members of the disjoint-classes construct are not skolemized. See Figure 9 for an example of N-Triples encoding of disjoint subclasses using blank nodes and Skolemized IRIs.

```

<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2002/07/owl#Class> .
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/2000/01/rdf-schema#subClassOf>
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Aerodrome> .
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/2004/02/skos/core#altLabel>
"Airfield"@en .
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/2004/02/skos/core#altLabel>
"Airport"@en .
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/2004/02/skos/core#prefLabel> "Land
Aerodrome"@en .
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/2000/01/rdf-schema#isDefinedBy>
<http://nsgreg.nga.mil/as/view?i=100436> .
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/2004/02/skos/core#definition>
"Definition: An aerodrome on land intended to be used either wholly or in part for the arrival, departure and surface
movement of aircraft. Description: [None Specified]"@en .
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> <http://www.w3.org/2000/01/rdf-schema#label>
"LandAerodrome"@en .

```

Figure 8 – N-Triples Encoding: Entity Class LandAerodrome

```

<http://api.nsgreg.nga.mil/.well-known/genid/2F0369E712864B0F9A76B91C650E619D> <http://www.w3.org/1999/02/22-rdf-
syntax-ns#type> <http://www.w3.org/2002/07/owl#AllDisjointClasses> .
<http://api.nsgreg.nga.mil/.well-known/genid/2F0369E712864B0F9A76B91C650E619D>
<http://www.w3.org/2002/07/owl#members> _:GCSR7AB2D77100804C3EA1F2962AEC5CF4F4 .
_:GCSR7AB2D77100804C3EA1F2962AEC5CF4F4 <http://www.w3.org/1999/02/22-rdf-syntax-ns#first>
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#Heliport> .
_:GCSR7AB2D77100804C3EA1F2962AEC5CF4F4 <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest>
_:GCSR97EDAE18891C41F59D005997BD82FD9A .
_:GCSR97EDAE18891C41F59D005997BD82FD9A <http://www.w3.org/1999/02/22-rdf-syntax-ns#first>
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#LandAerodrome> .
_:GCSR97EDAE18891C41F59D005997BD82FD9A <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest>
_:GCSR05E1A808C50F4766A539DA81B44ED065 .
_:GCSR05E1A808C50F4766A539DA81B44ED065 <http://www.w3.org/1999/02/22-rdf-syntax-ns#first>
<http://api.nsgreg.nga.mil/ontology/neo-ent/1-3#WaterAerodrome> .
_:GCSR05E1A808C50F4766A539DA81B44ED065 <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest>
<http://www.w3.org/1999/02/22-rdf-syntax-nil> .

```

Figure 9 – N-Triples Encoding: Disjoint Subclasses of Aerodrome

6 Governance and Publication

6.1 Introduction

The NEO Standard and its associated NEO content shall be governed and published in accordance with the general process established by the NGA as a Standards Development Organization (SDO) under the Functional Manager for GEOINT. These processes are currently executed by the Geospatial Intelligence (GEOINT) Content Standards Board (GCSB). The organization of the GCSB is described in the *GCSB Operations Guide* (available online).⁴²

6.2 Governance

The management of this NEO Standard conforms to the governance process established by NGA as an SDO under the Functional Manager for GEOINT.

The Geospatial Intelligence (GEOINT) Content Standards Board (GCSB) is the community forum responsible for providing governance, community coordination, prioritization of content development, and notifications for the set of NGA-developed GEOINT Data Standards that define a common method for specifying and encoding geospatial intelligence and related geospatial information in the NSG. Changes to the NEO Standard and its associated NEO content shall conform to the current governance process as described in the *GCSB Operations Guide*.

The NEO Standard and its associated NEO content evolve in response to NSG community requirements.⁴³ The GCSB is responsible for approving changes, distributing change notifications, and publishing the NEO Standard and NEO content for use by the U.S. Department of Defense (DoD), U.S. Intelligence Community (IC), and U.S. civil federal agencies. The NEO Standard and technical artifacts containing NEO content are published in the NSG-unique Standards Register of the NSG Standards Registry (NSGREG). The NEO content is also accessible through the REST API component of the NSGREG.

6.3 Publication

6.3.1 Introduction

The process for the publication of the NEO Standard and associated NEO content is described in Section 2.3.5 (Implementation of Changes) of the *GCSB Operations Guide*. The NEO Standard is published in the NSG-unique Standards Register of the NSGREG, at <http://nsgreg.nga.mil/doc/view?i=2615>.

The managed NEO content consists of:

- 1) Technical artifacts: RDF/XML and N-Triples encodings published in the NSGREG (<http://nsgreg.nga.mil/doc/view?i=4380>); and
- 2) Online resources retrievable through the REST API component of the NSG Standards Registry:
 - o NEO entities: <http://api.nsgreg.nga.mil/ontology/neo-ent>
 - o NEO enumerations: <http://api.nsgreg.nga.mil/ontology/neo-enum>

Non-versioned IRIs return the latest content baselines. Specific baselines may be requested using versioned IRIs.

All official publications of NEO content shall conform to the information model specified in Section 5.2 of this NEO Standard. All encodings shall conform to the OWL 2 representation and encoding as specified in Sections 5.2.8.20 and 5.4. The publication of NEO content is described in the following sections.

6.3.2 Publication of NEO Content as a Technical Artifact

The NEO technical artifacts are registered files containing NEO content baselines in OWL 2, encoded as specified in Section 5.4 of this standard. Content baselines are designated by version numbers (e.g., '1-3'). The first integer in the version number indicates the edition of the Standard on which the content baseline is based, while the second integer indicates the specific baseline in a possible sequence of updated content baselines.

⁴² Available online from the NSG Standards Registry, at <http://nsgreg.nga.mil/doc/view?i=4284>.

⁴³ The NEO content is derived from the content of the NSG Application Schema (NAS) using the rule-based approach of ISO 19150-2 as implemented in the ShapeChange application.

For each content baseline of the NEO, two pairs of technical artifacts are published with the OWL 2 encoding of the NEO content baseline; one pair in OWL 2 RDF/XML format and one pair in OWL 2 N-Triples format. Each pair of files contains the complete NEO content as of the official date of the baseline; a pair includes one file that encodes the content of the versioned “neo-ent” namespace (NEO entities) and one file that encodes the content of the “neo-enum” namespace (NEO enumerations). The two pairs of encoding files are published in the NSG-unique Standards Register of the NSG Standards Registry. A list of content baselines with links to their encoded content files is provided at <http://nsgreg.nga.mil/neo>.

These encoding files are designed to be machine-processable semantic resources. Their content may be directly examined in a text editor; however, an ontology application capable of displaying OWL 2 with a graphical user interface may be used to inspect the NEO content in a more human-friendly manner. A sample use of an ontology viewer for that purpose is presented in Annex D (informative).

6.3.3 Publication of NEO Content as REST API-accessible Resources

The NEO content is also available through the REST API component of the NSG Standards Registry. Files containing the latest NEO content baseline are retrievable from the following non-versioned IRIs:

- <http://api.nsgreg.nga.mil/ontology/neo-ent>
- <http://api.nsgreg.nga.mil/ontology/neo-enum>

The NEO landing page (<http://nsgreg.nga.mil/neo>) publishes a table listing each NEO content baseline with its versioned IRI. The versioned IRI (see Section 5.4.2.4) shall be used for authoritative citation of the NEO content in information exchange and data sharing. The versioned IRI shall also be used for official specification of the NEO content baseline version to be used in systems development and acquisition.

HTTP content negotiation based on the Accept request-header field may be used to specify the media type as RDF/XML ('application/rdf+xml') or N-Triples format ('application/n-triples') when resource retrieval is requested.⁴⁴ A request that does not specify a media type retrieves files in the default format (RDF/XML).

In addition, each NEO enumeration and listed value is also published separately as a resource accessible through the REST API component of the NSG Standards Registry. These resources are provided to support reference and re-use by Web-enabled applications in specifying data values.⁴⁵

Each resource representing an EnumeratedType (Section 5.4.4) shall include the following components based on the NEO information model:

- the OWL class representing the enumerated type;
- the SKOS Concept Scheme representing the enumerated type;
- SKOS Concepts representing all the listed values belonging to the enumerated type;
- a Different Terms assertion declaring the distinctness of all the included listed values.

Each SKOS Concept Scheme representing an EnumeratedType shall also be published as a REST API-accessible resource including the member listed values as SKOS Concepts, with the appropriate Different Terms assertion.

Each resource for a listed value shall be published as a REST API-accessible resource including only the documentation and properties for the SKOS Concept representing the listed value.

All separately published resources for NEO components shall include a `dct:isPartOf` assertion indicating that the resource represents partial content of the “neo-enum” namespace.

The external codelists referenced by NEO from the Information Resources (IR) Registry of the NSG Standards Registry follow the same pattern (with a different URI base) and return resources structured in the same way. The URI base for IR codelists referenced by NEO is: <http://api.nsgreg.nga.mil/codelist>.

Sample contents of REST API-accessible resource files with encodings for the enumeration class `ApronAccessibilityStatusType`, its associated concept scheme (`ApronAccessibilityStatusType_ConceptScheme`), and two of its listed values (a top concept and a subordinate concept) are presented in the figures below.

⁴⁴ For example, the request for the latest encoding of the 'neo-ent' file in N-Triples format is: <http://api.nsgreg.nga.mil/ontology/neo-ent?accept=application/n-triples>.

⁴⁵ Other types of ontology concepts are not offered as REST API-accessible resources and should be retrieved with the 'neo-ent' and 'neo-enum' baseline requests.

```

<rdf:RDF
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:e="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  <owl:Class rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType">
    <owl:oneOf rdf:parseType="Collection">
      <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/closed">
        <skos:topConceptOf>
          <skos:ConceptScheme rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme">
            <skos:prefLabel xml:lang="en">Apron Accessibility Status Type - Concept Scheme</skos:prefLabel>
            <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=106468"/>
            <skos:definition xml:lang="en">&lt;![CDATA[Definition: A coded domain value denoting the accessibility status type of an apron. Description: [None Specified]]&gt;&gt;</skos:definition>
            <rdfs:label xml:lang="en">ApronAccessibilityStatusType_ConceptScheme</rdfs:label>
            <dct:isFormatOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType"/>
            <dct:isPartOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3"/>
            </skos:ConceptScheme>
          </skos:topConceptOf>
          <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
          <skos:prefLabel xml:lang="en">Closed (Apron Accessibility Status Type)</skos:prefLabel>
          <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116678"/>
          <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is officially prohibited. Description: May be covered and/or blocked by a physical barrier.]]&gt;&gt;</skos:definition>
          <rdfs:label xml:lang="en">closed</rdfs:label>
        </e:ApronAccessibilityStatusType>
      <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/limited">
        <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
        <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
        <skos:prefLabel xml:lang="en">Limited (Apron Accessibility Status Type)</skos:prefLabel>
        <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116677"/>
        <skos:definition xml:lang="en">&lt;![CDATA[Definition: A limitation on access, but not function, has been imposed. Description: Not necessarily enforced by a physical barrier.]]&gt;&gt;</skos:definition>
        <rdfs:label xml:lang="en">limited</rdfs:label>
      </e:ApronAccessibilityStatusType>
      <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked">
        <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
        <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
        <skos:prefLabel xml:lang="en">Locked (Apron Accessibility Status Type)</skos:prefLabel>
      </e:ApronAccessibilityStatusType>
    </owl:oneOf>
  </owl:Class>

```

```

    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116672"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is prevented by a physical barrier, requiring special means to pass (for example: a key).
Description: [None Specified]]&gt;</skos:definition>
    <rdfs:label xml:lang="en">locked</rdfs:label>
  </e:ApronAccessibilityStatusType>
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedClosed">
    <skos:broader rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:prefLabel xml:lang="en">Locked Closed (Apron Accessibility Status Type)</skos:prefLabel>
    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116674"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is officially prohibited and is restricted by a physical barrier, requiring special means to
pass (for example: a key). Description: [None Specified]]&gt;</skos:definition>
    <rdfs:label xml:lang="en">lockedClosed</rdfs:label>
  </e:ApronAccessibilityStatusType>
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedOpen">
    <skos:broader rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:prefLabel xml:lang="en">Locked Open (Apron Accessibility Status Type)</skos:prefLabel>
    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116675"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is officially allowed although restricted by a physical barrier that is currently open, requiring
special means to close and prevent future passage (for example: a key). Description: [None Specified]]&gt;</skos:definition>
    <rdfs:label xml:lang="en">lockedOpen</rdfs:label>
  </e:ApronAccessibilityStatusType>
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/open">
    <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:prefLabel xml:lang="en">Open (Apron Accessibility Status Type)</skos:prefLabel>
    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116673"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is officially allowed. Description: May be covered and/or blocked by a physical barrier that
is temporarily passable.]]&gt;</skos:definition>
    <rdfs:label xml:lang="en">open</rdfs:label>
  </e:ApronAccessibilityStatusType>
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/restricted">
    <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:prefLabel xml:lang="en">Restricted (Apron Accessibility Status Type)</skos:prefLabel>
    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116676"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is officially allowed although a limitation on function has been imposed. Description: Not
necessarily enforced by a physical barrier.]]&gt;</skos:definition>
    <rdfs:label xml:lang="en">restricted</rdfs:label>
  </e:ApronAccessibilityStatusType>
</owl:oneOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
<skos:prefLabel xml:lang="en">Apron Accessibility Status Type</skos:prefLabel>
<rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=106468"/>

```

```

<skos:definition xml:lang="en">&lt;![CDATA[Definition: A coded domain value denoting the accessibility status type of an apron. Description: [None Specified]]]&gt;</skos:definition>
<rdfs:label xml:lang="en">ApronAccessibilityStatusType</rdfs:label>
<dct:isPartOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3"/>
</owl:Class>
<owl:AllDifferent>
<owl:distinctMembers rdf:parseType="Collection">
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/closed"/>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/limited"/>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked"/>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedClosed"/>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedOpen"/>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/open"/>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/restricted"/>
</owl:distinctMembers>
</owl:AllDifferent>
</rdf:RDF>

```

Figure 10 – Resource Representation for NEO Enumeration ApronAccessibilityStatusType

```

<rdf:RDF
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:e="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <skos:ConceptScheme rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme">
    <skos:prefLabel xml:lang="en">Apron Accessibility Status Type - Concept Scheme</skos:prefLabel>
    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=106468"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: A coded domain value denoting the accessibility status type of an apron. Description: [None Specified]]]&gt;</skos:definition>
    <rdfs:label xml:lang="en">ApronAccessibilityStatusType_ConceptScheme</rdfs:label>
    <dct:isFormatOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType"/>
    <dct:isPartOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3"/>
  </skos:ConceptScheme>
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/closed">
    <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
  </e:ApronAccessibilityStatusType>
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/limited">
    <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
  </e:ApronAccessibilityStatusType>

```

```

</e:ApronAccessibilityStatusType>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked">
  <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
  <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
</e:ApronAccessibilityStatusType>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedClosed">
  <skos:broader rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme/locked"/>
  <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
</e:ApronAccessibilityStatusType>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedOpen">
  <skos:broader rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme/locked"/>
  <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
</e:ApronAccessibilityStatusType>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/open">
  <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
  <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
</e:ApronAccessibilityStatusType>
<e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/restricted">
  <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
  <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
</e:ApronAccessibilityStatusType>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/closed"/>
    <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/limited"/>
    <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked"/>
    <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedClosed"/>
    <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedOpen"/>
    <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/open"/>
    <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/restricted"/>
  </owl:distinctMembers>
</owl:AllDifferent>
</rdf:RDF>

```

Figure 11 – Resource Representation for NEO Enumeration ApronAccessibilityStatusType_ConceptScheme

```

<rdf:RDF
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:e="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked">
    <skos:topConceptOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:prefLabel xml:lang="en">Locked (Apron Accessibility Status Type)</skos:prefLabel>
    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116672"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is prevented by a physical barrier, requiring special means to pass (for example: a key).]]&gt;</skos:definition>
  ). Description: [None Specified]]&gt;</skos:definition>
    <rdfs:label xml:lang="en">locked</rdfs:label>
    <dct:isPartOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3"/>
  </e:ApronAccessibilityStatusType>
</rdf:RDF>

```

Figure 12 – Resource Representation for NEO Listed Value ApronAccessibilityStatusType/locked

```

<rdf:RDF
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:e="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <e:ApronAccessibilityStatusType rdf:about="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/lockedOpen">
    <skos:broader rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType/locked"/>
    <skos:inScheme rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3/ApronAccessibilityStatusType_ConceptScheme"/>
    <skos:prefLabel xml:lang="en">Locked Open (Apron Accessibility Status Type)</skos:prefLabel>
    <rdfs:isDefinedBy rdf:resource="http://nsgreg.nga.mil/as/view?i=116675"/>
    <skos:definition xml:lang="en">&lt;![CDATA[Definition: Access is officially allowed although restricted by a physical barrier that is currently open, requiring special means to close and prevent future passage (for example: a key). Description: [None Specified]]&gt;&gt;</skos:definition>
    <rdfs:label xml:lang="en">lockedOpen</rdfs:label>
    <dct:isPartOf rdf:resource="http://api.nsgreg.nga.mil/ontology/neo-enum/1-3"/>
  </e:ApronAccessibilityStatusType>
</rdf:RDF>

```

Figure 13 – Resource Representation for NEO Listed Value ApronAccessibilityStatusType/lockedOpen

Annex A – Conformance (Normative)

A.1 Introduction

Conformance is the fulfilment of specified requirements.⁴⁶ Conformance to the NSG Enterprise Ontology (NEO) (including the NEO Standard and associated NEO content) shall be determined based on the tests specified in this Annex. Any product claiming conformance to the NEO shall pass all the requirements specified in the abstract test suite in Section A.2.

A general explanation of the approach to conformance testing is presented in this section, including relevant terminology. The conformance testing framework specified in Section A.2 is based on ISO 19105:2000 *Geographic information – Conformance and testing*. The definition of an abstract test suite for conformance testing appears in ISO 19105, together with an explanation of the testing framework. The format for conformance clauses is specified in ISO 19105, Annex A.

A.1.1 Terms and Definitions

A special terminology is used to describe the conformance testing framework. Terms and definitions⁴⁷ specific to this annex are presented in Table 23. Terms that are defined in ISO 19105:2000 have a number in parentheses referring to the clause of that standard in which the term is defined.

Table 23 – Terms and Definitions for Conformance Testing

Term	Definition
abstract test case (ATC)	A generalized test for a particular requirement. (3.1) NOTE: An abstract test case is a formal basis for deriving executable test cases. One or more test purposes are encapsulated in the abstract test case. An abstract test case is independent of both the implementation and the values. It should be complete in the sense that it is sufficient to enable a test verdict to be assigned unambiguously to each potentially observable test outcome (<i>i.e.</i> , sequence of test events).
abstract test module (ATM)	A set of related abstract test cases . (3.3) NOTE: Abstract test modules may be nested in a hierarchical way.
abstract test suite (ATS)	An abstract test module specifying all the requirements to be satisfied for conformance . (3.4)
basic test	An initial capability test intended to identify clear cases of non-conformance. (3.6) NOTE: Basic tests may be used to determine whether to conduct further tests.
capability test	A test designed to determine whether an implementation under test conforms to a particular characteristic of a standard as described in the test purpose. (3.7) NOTE: Capability tests check that the capabilities claimed in an implementation conformance statement (ICS) are consistent with the observable capabilities of the implementation under test.
conformance	The fulfilment of specified requirements. (3.8) NOTE: Conformance may be claimed for any product , <i>i.e.</i> , data or software or services or for specifications including any profile or functional standard.
conformance testing	The testing of a product to determine the extent to which the product is a conforming implementation . (3.11)

⁴⁶ ISO 19105:2000 *Geographic information – Conformance and testing*.

⁴⁷ In the definitions in Table 23, a term is styled in **bold** when the meaning of that term is specified elsewhere in the table.

contains	Includes a representation of the content of the ontology. NOTE: A product may contain the NEO content either directly (for example, by importing the RDF/XML encoding of the NEO content), or by reference (for example, using the IRI of NEO components).
implementation	A realization of a specification. (3.18) NOTE1: In the context of the ISO geographic information standards, this includes specifications of geographic information services and datasets. (3.18) NOTE2: An implementation under test (IUT) is a product being evaluated (e.g., by conformance testing or performance testing) according to identified criteria. (3.24)
implementation conformance statement (ICS)	A statement of the options which have been implemented. (3.19) NOTE: This will allow the implementation to be tested for conformance against the relevant requirements, and against those requirements only. This statement shall contain only options within the framework of requirements specified in the relevant geographic information standards.
product	Data or software or a service. (3.18 NOTE) NOTE: A candidate product is a product submitted for conformance testing .
verdict	The result of a test. (6.4.4) NOTE: The value of a test verdict is one of: <i>pass</i> , <i>fail</i> , or <i>inconclusive</i> . Verdict criteria are specified by an abstract test case .

A.1.2 Conformance Testing Methodology

Conformance testing for the NEO is specified by this abstract test suite (ATS).

An ATS comprises all the abstract test cases needed to produce an overall verdict about the conformance of a candidate product being considered as an implementation under test (IUT).⁴⁸ Abstract test cases may be collected in a set of related tests called an abstract test module. Abstract test modules may be nested. An abstract test suite includes test modules and other test cases arranged in a hierarchy of conformance tests.

Each abstract test case is designed to test a candidate product for conformance to a specific requirement. A test case has several components:

- a) A test-case identifier;
- b) A stated test purpose that is a precise description of the test objective and also indicates whether the requirement being tested is mandatory, conditional, or optional;
- c) A description of the test method, specifying the test criteria that shall be used to determine the test verdict. A test may evaluate a multi-part requirement. The method indicates the way in which the test shall be conducted (e.g., manual or automated). The test method may reference other clauses in the test suite.
- d) References to one or more sections in the standard that identify the requirements addressed by the test.
- e) The test type (either a basic test or a capability test).

Mandatory requirements are those which shall be observed in all cases. Conditional requirements shall be observed if the conditions set out in the specification apply. Optional requirements may be selected to suit the implementation, provided that any requirements applicable to the option are observed.⁴⁹

In addition to an ATS, testing requires an implementation conformance statement (ICS) that declares which capabilities have been implemented for the product. This is especially important when there are options that may be implemented (or not), in order to evaluate the conformance of a particular implementation against the relevant requirements.

⁴⁸ Examples of types of candidate products are listed in Section 2.1.

⁴⁹ ISO 19105:2000, Section 5.3.

Products that claim conformance to the NEO shall support the mandatory RDF/XML encoding of the NEO content as specified in the officially published technical artifacts. They also may support the optional N-Triples encoding of the NEO content.

The ATS for the NEO (Section A.2) specifies conformance evaluation of a product using the NEO content as a whole, that is, with all of the content specified in the (mandatory) RDF/XML encoding and the (optional) N-Triples encoding.

An implementation conformance statement (ICS) for a product to be tested for conformance to the NEO shall contain the following information regarding the capabilities that have been implemented for the product:

- I. Identification of the NEO content baseline to which the product conforms. Each baseline indicates the NEO Standard edition on which it is based.
- II. Statement of what encodings the product supports. (Note: The product shall conform to the RDF/XML encoding of the NEO content, and may also optionally conform to the N-Triples encoding.)
- III. Statement of whether the product uses NEO content via active IRI-based Web links (using the REST API component of the NSG Standards Registry) or from locally installed copies of the officially published technical artifacts.
- IV. Statement that the product conforms to the NEO content in full.
- V. Explanation of how to acquire authorized access to the system(s) where the product is installed, if needed to test the product.

Abstract test cases may be automated for performance by a software system. Manual testing may be necessary when human judgment is required or when automated testing is too complex.

A.1.3 Logical Structure of the Abstract Test Suite

The abstract test suite for the NEO contains three top-level test modules, each of which contains multiple test modules and/or test cases. The structure of the test suite is depicted in Figure 14.

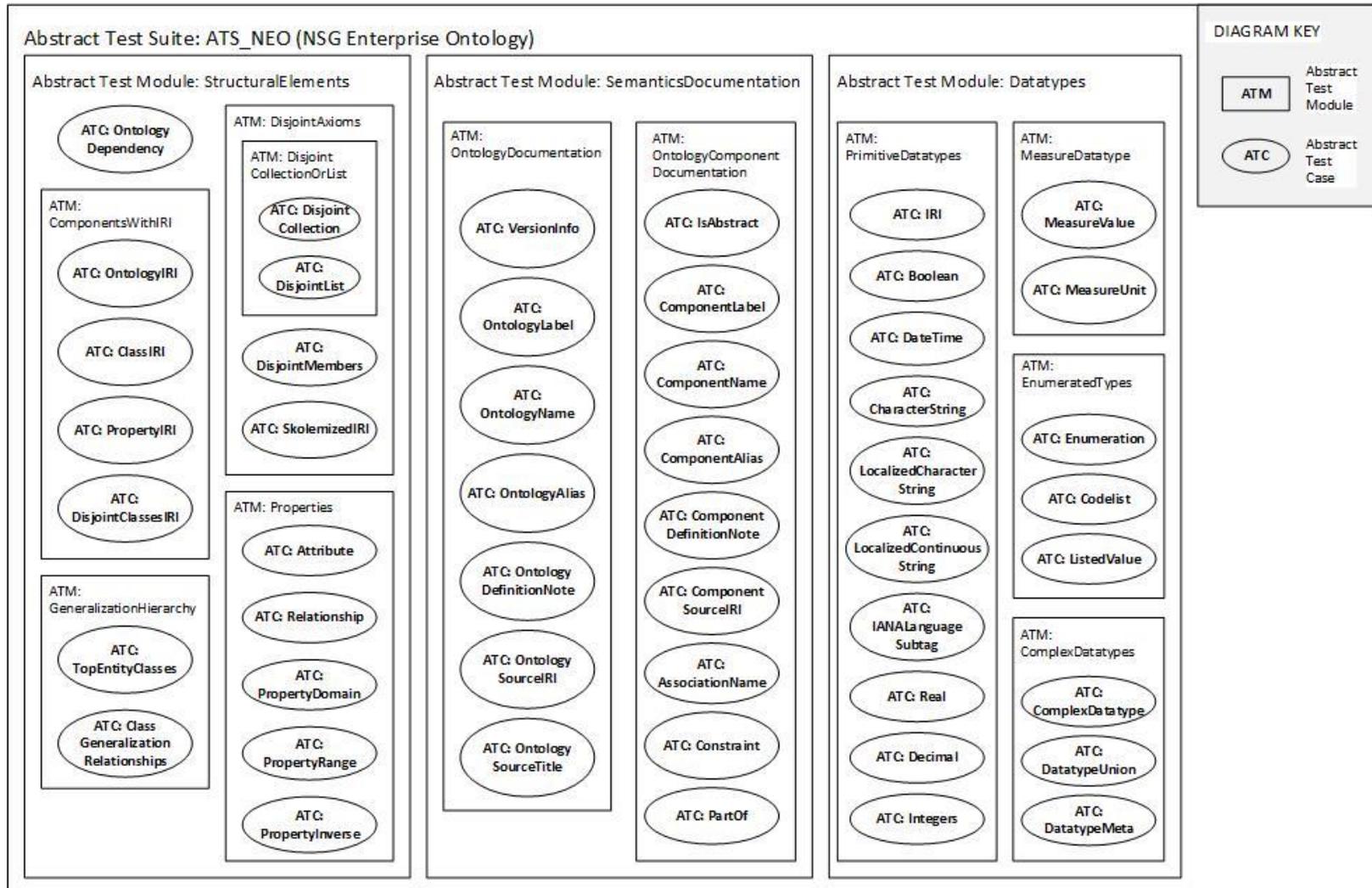


Figure 14 – Structure of the Abstract Test Suite for the NSG Enterprise Ontology

A.2 **Abstract Test Suite for the NSG Enterprise Ontology (NEO)**

- a) Test identifier: ATS_NEO
- b) Test purpose: Verify the conformance of the product with the NEO (including the NEO Standard and associated NEO content). **(Mandatory)**
- c) Test method: Inspect the product to determine that it contains the required NEO structural elements (A.2.1), semantics documentation (A.2.2), and datatypes (A.2.3), in accordance with the requirements in the NEO (including the NEO Standard and associated NEO content).
- d) Reference: NEO Standard, Section 5; NEO content baseline identified in the ICS.
- e) Test type: Basic

NOTE: If an information construct from the NEO Standard is employed within a product, then the meaning and structure of that construct shall be preserved, and information regarding the corresponding construct shall be exactly as specified in the NEO Standard. If NEO content is employed within a product, then the meaning and structure of the content shall be consistent with the NEO content as officially published in the mandatory RDF/XML encoding.

A.2.1 **Test Module for Conformance to Ontology Structure**

- a) Test identifier: StructuralElements
- b) Test purpose: Verify the conformance of the product with the required NEO structural elements. **(Mandatory)**
- c) Test method: Inspect the product to determine that it contains the required structural elements, including all ontology dependencies (A.2.1.1), elements with identity (A.2.1.2), generalization hierarchy (A.2.1.3), disjoint-classes axioms (A.2.1.4), and property declarations (A.2.1.5).
- d) Reference: NEO Standard (Figure 1; Sections 5.2.3, 5.2.4, 5.2.5, and 5.2.6; Sections 5.4.3.2, 5.4.3.3, 5.4.3.4, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

A.2.1.1 **Test Case for Ontology Dependency(ies)**

- a) Test identifier: OntologyDependency
- b) Test purpose: Verify the conformance of the product with dependencies asserted in the NEO content. **(Mandatory)**
- c) Test method: Inspect the product to determine that it contains all of the `owl:imports` that are declared in the NEO content.
- d) Reference: NEO Standard (Sections 5.2.3 and 5.4.3.2).
- e) Test type: Basic

A.2.1.2 **Test Module for Components with Identity (IRIs)**

- a) Test identifier: ComponentsWithIRI
- b) Test purpose: Verify the conformance of the product with all of the specified NEO content having identity indicated by IRIs in the NEO namespace. **(Mandatory)**
- c) Test method: Inspect the product to determine that it contains all of the NEO content having IRI values in the NEO namespace, including the Ontology itself (A.2.1.2.1), EntityClasses (A.2.1.2.2), Properties (A.2.1.2.3), and (if applicable) DisjointClasses (A.2.1.2.4).
- d) Reference: NEO Standard (Sections 5.4.2.4 and 5.4.2.5).
- e) Test type: Basic

A.2.1.2.1 **Test Case for Ontology with IRI**

- a) Test identifier: OntologyIRI

- b) Test purpose: Verify the conformance of the product with the Ontology declaration and IRI value required for the indicated NEO content baseline. **(Mandatory)**
- c) Test method: (1) Inspect the product in order to determine that it contains a declaration of the NEO as an `owl:Ontology`, in which the value in `rdf:about` is the IRI of the NEO content baseline indicated in the implementation conformance statement (ICS). (2) Inspect the product in order to determine that it contains an `owl:versionIRI` declaration in which the value is the IRI of the NEO content baseline indicated in the ICS. The two values for the content baseline IRI shall be identical. The IRI for the NEO shall always identify the applicable content baseline by utilizing an IRI that indicates the NEO version as identified in the ICS.
- d) Reference: NEO Standard (Section 5.4.2.4).
- e) Test type: Basic

A.2.1.2.2 Test Case for Entity Classes with IRIs

- a) Test identifier: ClassIRI
- b) Test purpose: Verify the conformance of the product with the EntityClass declarations and IRI values required for each EntityClass in the NEO content baseline indicated in the ICS. **(Mandatory)**
- c) Test method: Inspect the product in order to determine (1) that it contains an `owl:Class` declaration for each EntityClass in the NEO content baseline indicated in the ICS, and (2) that the value in `rdf:about` for each EntityClass is the value of the classIRI in the NEO content. The IRI for a NEO EntityClass shall always identify the applicable content baseline by utilizing an IRI that indicates the NEO version as identified in the ICS.
- d) Reference: NEO Standard (Section 5.4.2.5).
- e) Test type: Basic

A.2.1.2.3 Test Case for Entity Properties with IRIs

- a) Test identifier: PropertyIRI
- b) Test purpose: Verify the conformance of the product with the EntityProperty (specifically, EntityAttribute and EntityRelationship) declarations and IRI values required for each EntityProperty in the NEO content baseline indicated in the ICS. **(Mandatory)**
- c) Test method: Inspect the product in order to determine (1) that it contains an `owl:DatatypeProperty` or an `owl:ObjectProperty` declaration for each EntityProperty in the NEO content baseline indicated in the ICS, and (2) that the value in `rdf:about` for each EntityAttribute or EntityRelationship is the value of the propertyIRI in the NEO content. The IRI for a NEO EntityProperty shall always identify the applicable content baseline by utilizing an IRI that indicates the NEO version as identified in the ICS.
- d) Reference: NEO Standard (Section 5.4.2.5).
- e) Test type: Basic

A.2.1.2.4 Test Case for DisjointClasses with IRIs

- a) Test identifier: DisjointClassesIRI
- b) Test purpose: Verify the conformance of the product with the declaration of DisjointClasses using Skolemized IRIs. (*Conditional* on the implementation being in the N-Triples encoding)
- c) Test method: See A.2.1.4.3.
- d) Reference: NEO Standard (Section 5.4.3.4).
- e) Test type: Capability test

A.2.1.3 Test Module for Generalization (Subclass) Hierarchy

- a) Test identifier: GeneralizationHierarchy
- b) Test purpose: Verify the conformance of the product with the complete class-generalization hierarchy (*i.e.*, subclass tree) of the NEO content. **(Mandatory)**

- c) Test method: Inspect the product in order to determine that it contains all of the most general concepts (A.2.1.3.1) in the NEO content and all of the `rdfs:subClassOf` relationships (A.2.1.3.2) between EntityClasses in the NEO content.
- d) Reference: NEO Standard (Section 5.3.5 and Section 5.4.3.3).
- e) Test type: Basic

A.2.1.3.1 Test Case for Top Entity Classes

- a) Test identifier: TopEntityClasses
- b) Test purpose: Verify the conformance of the product with the most basic (*i.e.*, top) EntityClasses in the NEO content. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains all of the most general EntityClasses in the NEO content, *i.e.*, the EntityClasses that are not subclasses of any other NEO EntityClass.
- d) Reference: NEO Standard (Section 5.4.3.3).
- e) Test type: Basic

A.2.1.3.2 Test Case for Generalization Relationships

- a) Test identifier: GeneralizationRelationships
- b) Test purpose: Verify the conformance of the product with all the generalization relationships in the NEO content. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains all of the `rdfs:subClassOf` relationships that are declared between EntityClasses in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.3).
- e) Test type: Basic

A.2.1.4 Test Module for Disjointness Axioms

- a) Test identifier: DisjointAxioms
- b) Test purpose: Verify the conformance of the product with the requirement to represent the disjointness constraints on all sets of sibling EntityClasses in the NEO content. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required DisjointClasses axioms (A.2.1.4.1) used for declaring the pairwise disjointness of sibling EntityClasses (A.2.1.4.2) in the NEO content, with Skolemized IRIs (A.2.1.4.3) where required.
- d) Reference: NEO Standard (Section 5.4.3.4, Section 5.4.5.2, and Section 5.4.5.3).
- e) Test type: Basic

NOTE: Sibling EntityClasses are those which have the same EntityClass as their generalization (*i.e.*, their superclass).

A.2.1.4.1 Test Module for Disjoint Collection or List

- a) Test identifier: DisjointCollectionOrList
- b) Test purpose: Verify the conformance of the product with the requirement to include components to represent DisjointClasses axioms for all sets of sibling EntityClasses. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that (1) it contains the required structural components appropriate to the encoding (A.2.1.4.1.1 or A.2.1.4.1.2) for stating the DisjointClasses axioms for sibling subclasses, and (2) it contains a DisjointClasses component for every set of sibling EntityClasses in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.4, Section 5.4.5.2, and Section 5.4.5.3).
- e) Test type: Basic

A.2.1.4.1.1 Test Case for DisjointClasses Collection

- a) Test identifier: DisjointCollection
- b) Test purpose: Verify the conformance of the product with the requirement to represent DisjointClasses as collections of EntityClasses. (*Conditional* on the use of the (mandatory) RDF/XML encoding in the IUT)
- c) Test method: Inspect the product in order to determine that it contains the required `owl:AllDisjointClasses` component, declared as a (RDF `parseType`) collection, for each set of sibling EntityClasses in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.4 and Section 5.4.5.2).
- e) Test type: Capability test

A.2.1.4.1.2 Test Case for DisjointClasses List

- a) Test identifier: DisjointList
- b) Test purpose: Verify the conformance of the product with the requirement to represent DisjointClasses as lists of EntityClasses. (*Conditional* on the use of the (optional) N-Triples encoding in the IUT)
- c) Test method: Inspect the product in order to determine that it contains the required <http://www.w3.org/2002/07/owl#AllDisjointClasses> component, structured as an RDF List, for each set of sibling EntityClasses in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.4 and Section 5.4.5.3).
- e) Test type: Capability test

A.2.1.4.2 Test Case for Members of DisjointClasses

- a) Test identifier: DisjointMembers
- b) Test purpose: Verify the conformance of the product with the required enumeration of all sibling EntityClasses within a DisjointClasses axiom. (**Mandatory**)
- c) Test method: Inspect the product in order to determine that it contains (1) (for the RDF/XML encoding) a complete collection of the `owl:members` of each `owl:AllDisjointClasses` component in the NEO content, or (2) (for the N-Triples encoding) a complete list of the <http://www.w3.org/2002/07/owl#members> of each <http://www.w3.org/2002/07/owl#AllDisjointClasses> component in the NEO content.
- d) Reference: NEO Standard (Section 5.4.5.2 and Section 5.4.5.3).
- e) Test type: Capability test

A.2.1.4.3 Test Case for Skolemized IRIs

- a) Test identifier: SkolemizedIRI
- b) Test purpose: Verify the conformance of the product with the use of a Skolemized IRI required as the identifier for each representation of a DisjointClasses axiom. (*Conditional* on the use of the (optional) N-Triples encoding in the IUT)
- c) Test method: Inspect the product in order to determine that it contains a Skolemized IRI as the identifier for each instance of <http://www.w3.org/2002/07/owl#AllDisjointClasses> in the N-Triples encoding of the NEO content.
- d) Reference: NEO Standard (Section 5.4.5.3).
- e) Test type: Capability

NOTE: The URI base for a Skolemized IRI differs from the URI base of the NEO (e.g., "http://api.nsgreg.nga.mil/ontology/neo-ent/..."). Example of a Skolemized IRI: <http://api.nsgreg.nga.mil/well-known/genid/C26B41F050384C878C00D7D462C2730F>.

A.2.1.5 Test Module for Properties

- a) Test identifier: Properties

- b) Test purpose: Verify the conformance of the product with the encodings for each EntityAttribute and EntityRelationship in the indicated NEO content baseline. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the axioms used to define each Property in the NEO content, including whether it is an `owl:DatatypeProperty` or `owl:ObjectProperty` (A.2.1.5.1 and A.2.1.5.2), that property domains (A.2.1.5.3) and ranges (A.2.1.5.4) are declared where required, and that property inverses (A.2.1.5.5) are declared where required.
- d) Reference: NEO Standard (Section 5.4.3.5 and 5.4.3.6).
- e) Test type: Basic

A.2.1.5.1 Test Case for Entity Attributes

- a) Test identifier: Attribute
- b) Test purpose: Verify the conformance of the product with the declarations for each EntityAttribute in the indicated NEO content baseline. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains each EntityAttribute (represented as an `owl:DatatypeProperty` or `owl:ObjectProperty`) included in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.5).
- e) Test type: Basic

A.2.1.5.2 Test Case for Entity Relationships

- a) Test identifier: Relationship
- b) Test purpose: Verify the conformance of the product with the specified declarations for each EntityRelationship in the NEO content. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains each EntityRelationship (represented as an `owl:ObjectProperty`) in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.6).
- e) Test type: Basic

A.2.1.5.3 Test Case for Property Domain

- a) Test identifier: PropertyDomain
- b) Test purpose: Verify the conformance of the product with the declaration of a domain for each OWL Property representing an EntityAttribute or EntityRelationship. **(Mandatory)**
- c) Test method: Inspect the product to determine that, if the NEO content contains an (optional) `rdfs:domain` declaration for an OWL Property, then the product also contains that `rdfs:domain` declaration.
- d) Reference: NEO Standard (Section 5.4.3.5 and Section 5.4.3.6).
- e) Test type: Basic

A.2.1.5.4 Test Case for Property Range

- a) Test identifier: PropertyRange
- b) Test purpose: Verify the conformance of the product with the declaration of a range for each OWL Property representing an EntityAttribute or EntityRelationship. **(Mandatory)**
- c) Test method: Inspect the product to determine that, if the NEO content contains an (optional) `rdfs:range` declaration for an OWL Property, then the product also contains that `rdfs:range` declaration.
- d) Reference: NEO Standard (Section 5.4.3.5 and Section 5.4.3.6).
- e) Test type: Basic

A.2.1.5.5 Test Case for Property Inverse

- a) Test identifier: PropertyInverse
- b) Test purpose: Verify the conformance of the product with the declaration of an inverse relationship between OWL Object Properties where required. **(Mandatory)**
- c) Test method: Inspect the product to determine that, if the NEO content contains an `owl:inverseOf` declaration between two OWL Object Properties, then the product also contains that `owl:inverseOf` declaration.
- d) Reference: NEO Standard (Section 5.4.3.6).
- e) Test type: Basic

A.2.2 Test Module for Documentation of Semantics

- a) Test identifier: SemanticsDocumentation
- b) Test purpose: Verify the conformance of the product with the documentation properties in the NEO content (including both neo-ent and neo-enum namespaces) and the ontology components (Entity Classes and Properties), as required by the NEO Standard and associated NEO content. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains all of the required documentation properties with the correct values for the NEO ontologies (A.2.2.1) and ontology components (A.2.2.2).
- d) Reference: NEO Standard (Sections 5.4.3.2, 5.4.3.3, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

A.2.2.1 Test Module for Ontology Documentation

- a) Test identifier: OntologyDocumentation
- b) Test purpose: Verify the conformance of the product with the ontology documentation properties in the NEO content. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required ontology documentation properties with the correct values (A.2.2.1.1, A.2.2.1.2, A.2.2.1.3, A.2.2.1.4, A.2.2.1.5, A.2.2.1.6, and A.2.2.1.7) for the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).
- e) Test type: Basic

A.2.2.1.1 Test Case for Ontology Version Information

- a) Test identifier: VersionInfo
- b) Test purpose: Verify the conformance of the product with documentation of the required version information for the NEO content. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required documentation property `owl:versionInfo` with the value specified in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).
- e) Test type: Basic

A.2.2.1.2 Test Case for Ontology Label

- a) Test identifier: OntologyLabel
- b) Test purpose: Verify the conformance of the product with the documentation of the label for the NEO ontology. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required documentation `rdfs:label` with the value as specified in the NEO Standard and NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).

- e) Test type: Basic

A.2.2.1.3 Test Case for Ontology Name

- a) Test identifier: OntologyName
- b) Test purpose: Verify the conformance of the product with documentation of the name for the NEO ontology. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required documentation property `skos:prefLabel` with the value as specified in the NEO Standard and the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).
- e) Test type: Basic

A.2.2.1.4 Test Case for Ontology Alias

- a) Test identifier: OntologyAlias
- b) Test purpose: Verify the conformance of the product with documentation of alias(es) for the NEO ontology. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required documentation property `skos:altLabel` with the value "NEO" as specified in the NEO Standard and the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).
- e) Test type: Basic

A.2.2.1.5 Test Case for Ontology Definition Note

- a) Test identifier: OntologyDefinitionNote
- b) Test purpose: Verify the conformance of the product with documentation of the required definitionNote for the NEO ontology. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required documentation property `skos:definition` with the value as specified in the NEO Standard and the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).
- e) Test type: Basic

A.2.2.1.6 Test Case for Ontology Source Reference

- a) Test identifier: OntologySourceIRI
- b) Test purpose: Verify the conformance of the product with documentation of the required sourceIRI for the NEO ontology. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required documentation property `rdfs:isDefinedBy` with the value specified in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).
- e) Test type: Basic

A.2.2.1.7 Test Case for Ontology Source Title

- a) Test identifier: OntologySourceTitle
- b) Test purpose: Verify the conformance of the product with the documentation of the title of the Standard document on which the NEO is based. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it contains the required documentation property `dct:source` with the value specified in the NEO content.
- d) Reference: NEO Standard (Section 5.4.3.2).
- e) Test type: Basic

A.2.2.2 Test Module for Ontology Component Documentation

- a) Test identifier: `OntologyComponentDocumentation`
- b) Test purpose: Verify the conformance of the product with the required documentation properties for the `EntityClasses` and `EntityProperties` specified in the NEO content. (**Mandatory**)
- c) Test method: Inspect the product in order to determine that it contains the applicable `EntityClass` and `EntityProperty` documentation properties with the correct values (A.2.2.2.1, A.2.2.2.2, A.2.2.2.3, A.2.2.2.4, A.2.2.2.5, A.2.2.2.6, A.2.2.2.7, A.2.2.2.8, and A.2.2.2.9) for each `EntityClass` and `EntityProperty` in the NEO content.
- d) Reference: NEO Standard (Section 5.2.7, Section 5.3.6, Section 5.4.2.5, and Sections 5.4.3.3, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

A.2.2.2.1 Test Case for Abstract Ontology Component

- a) Test identifier: `IsAbstract`
- b) Test purpose: Verify the conformance of the product with the required indication of all abstract `EntityClasses`. (*Conditional* on the ontology component being an `EntityClass`)
- c) Test method: Inspect the product in order to determine that it contains an assertion of the Boolean documentation property `iso19150-2:isAbstract` with value `TRUE` for each abstract `EntityClass` in the NEO content.
- d) Reference: NEO Standard (Section 5.2.7, Section 5.3.6, and Section 5.4.3.3).
- e) Test type: Basic

NOTE: Abstract classes shall not be directly instantiated.

A.2.2.2.2 Test Case for Ontology Component Label

- a) Test identifier: `ComponentLabel`
- b) Test purpose: Verify the conformance of the product with the documentation of the label for every ontology component. (**Mandatory**)
- c) Test method: Inspect the product in order to determine that each ontology component has the required documentation property `rdfs:label` with the value specified in the NEO content. The value is the same as the terminal segment of the IRI for the component (*i.e.*, `ClassIRI` or `PropertyIRI`).
- d) Reference: NEO Standard (Section 5.4.2.5; Sections 5.4.3.3, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

A.2.2.2.3 Test Case for Ontology Component Name

- a) Test identifier: `ComponentName`
- b) Test purpose: Verify the conformance of the product with documentation of the name for each ontology component. (**Mandatory**)
- c) Test method: Inspect the product in order to determine that each Ontology component has the required documentation property `skos:prefLabel` with the value for the preferred human-readable name of the class as specified in the NEO content.
- d) Reference: NEO Standard (Section 5.2.7; Sections 5.4.3.3, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

NOTE: The preferred human-readable name of the Ontology component should be used to refer to that class in navigation menus, browse trees, or other displays.

A.2.2.2.4 Test Case for Ontology Component Alias

- a) Test identifier: `ComponentAlias`

- b) Test purpose: Verify the conformance of the product with documentation of the optional alias(es), if any, for each ontology component. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that each Ontology component has the documentation property `skos:altLabel` with the value(s) specified in the NEO content (if there are any). Aliases are optional elements in the NEO.
- d) Reference: NEO Standard (Section 5.2.7; Sections 5.4.3.3, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

A.2.2.2.5 Test Case for Ontology Component Definition Note

- a) Test identifier: ComponentDefinitionNote
- b) Test purpose: Verify the conformance of the product with documentation of the required definitionNote for each Ontology component. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that each ontology component has the required documentation property `skos:definition` with the value specified in the NEO content.
- d) Reference: NEO Standard (Section 5.2.7; Sections 5.4.3.3, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

A.2.2.2.6 Test Case for Ontology Component Source Reference

- a) Test identifier: ComponentSourceIRI
- b) Test purpose: Verify the conformance of the product with documentation of the required sourceIRI for each Ontology component. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that each Ontology component has the required documentation property `rdfs:isDefinedBy` with the value specified in the NEO content.
- d) Reference: NEO Standard (Section 5.2.7; Sections 5.4.3.3, 5.4.3.5, and 5.4.3.6).
- e) Test type: Basic

A.2.2.2.7 Test Case for Association Name

- a) Test identifier: AssociationName
- b) Test purpose: Verify the conformance of the product with the required documentation of the association name for each EntityRelationship component derived from an association role. (*Conditional* on the ontology component being an EntityRelationship derived from an association)
- c) Test method: Inspect the product in order to determine that each EntityRelationship component has the required documentation property `iso19150-2:associationName` as required, with the value specified in the NEO content.
- d) Reference: NEO Standard (Section 5.2.7; Section 5.4.3.6).
- e) Test type: Basic

A.2.2.2.8 Test Case for Constraint

- a) Test identifier: Constraint
- b) Test purpose: Verify the conformance of the product with the use of descriptions of constraints on EntityClass ontology components. (*Conditional* on the ontology component being an EntityClass that has a constraint)
- c) Test method: Inspect the product in order to determine that each EntityClass that has a constraint declaration specified in the NEO content has the documentation property `iso19150-2:constraint` with a natural language statement of the constraint, as specified in the NEO content.
- d) Reference: NEO Standard (Section 5.2.7; Section 5.4.3.3).
- e) Test type: Basic

A.2.2.2.9 Test Case for Ontology Component Part-of

- a) Test identifier: PartOf
- b) Test purpose: Verify the conformance of the product with the required use of `dct:isPartOf` declarations. (*Conditional* on a product that represents the ontology components in separate resource files)
- c) Test method: Inspect the product in order to determine that in each individual resource file, the Ontology component has the required documentation property `dct:isPartOf` with the value of the NEO IRI.
- d) Reference: NEO Standard (Section 6.3.3).
- e) Test type: Basic

A.2.3 Test Module for Datatype Conformance

- a) Test identifier: Datatypes
- b) Test purpose: Verify the conformance of the product with the datatypes specified in the NEO. (**Mandatory**)
- c) Test method: Inspect the product to determine that it uses the required datatypes and encodings for Primitive Datatypes (A.2.3.1), MeasureDatatype (A.2.3.2), EnumeratedTypes (A.2.3.3), and ComplexDatatypes (A.2.3.4), as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8; Section 5.4.4).
- e) Test type: Basic

A.2.3.1 Test Module for Primitive Datatypes

- a) Test identifier: PrimitiveDatatypes
- b) Test purpose: Verify the conformance of the product with the primitive datatypes specified in the NEO. (**Mandatory**)
- c) Test method: Inspect the product in order to determine that it uses primitive datatypes as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8; Section 5.4.4.2).
- e) Test type: Basic

A.2.3.1.1 Test Case for IRI Datatype

- a) Test identifier: IRI
- b) Test purpose: Verify the conformance of the product with the IRI datatype. (**Mandatory**)
- c) Test method: Inspect the product in order to determine that it uses values in the range of the required datatype `xsd:anyURI` for properties specified with the value type IRI in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.6; Section 5.4.4.2).
- e) Test type: Basic

A.2.3.1.2 Test Case for Boolean Datatype

- a) Test identifier: Boolean
- b) Test purpose: Verify the conformance of the product with the Boolean datatype. (**Mandatory**)
- c) Test method: Inspect the product in order to determine that it uses values in the range of the required datatype `xsd:boolean` for properties specified with the value type Boolean in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.5; Section 5.4.4.2).
- e) Test type: Basic

A.2.3.1.3 Test Case for DateTime Datatypes

- a) Test identifier: DateTime
- b) Test purpose: Verify the conformance of the product with the DateTime datatype. **(Mandatory)**
- c) Test method: Inspect the product to determine that it uses values that conform to the `xsd:dateTime` datatype for the properties specified with the value type DateTime in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.7; Section 5.4.4.2).
- e) Test type: Basic

A.2.3.1.4 Test Case for CharacterString Datatype

- a) Test identifier: CharacterString
- b) Test purpose: Verify the conformance of the product with the CharacterString datatype. **(Mandatory)**
- c) Test method: (1) Inspect the product in order to determine that it uses values in the range of the required datatype `rdf:PlainLiteral` for properties specified with the value type CharacterString in the NEO Standard and associated NEO content. (2) Values that are character strings (e.g., values of `xsd:string`) with no language tag satisfy the requirements for the value type CharacterString in the NEO model. (3) Optionally, a language tag may be present.
- d) Reference: NEO Standard (Section 5.2.8.2; Section 5.4.4.2).
- e) Test type: Basic

A.2.3.1.5 Test Case for LocalizedCharacterString Datatype

- a) Test identifier: LocalizedCharacterString
- b) Test purpose: Verify the conformance of the product with the LocalizedCharacterString datatype. **(Mandatory)**
- c) Test method: (1) Inspect the product in order to determine that it uses values in the range of the required datatype `rdf:PlainLiteral` for properties specified with the value type LocalizedCharacterString in the NEO Standard and associated NEO content. (2) In order to satisfy the requirements for a LocalizedCharacterString in the NEO information model, a value must include both a character string and a language tag.
- d) Reference: NEO Standard (Section 5.2.8.3; Section 5.4.4.2).
- e) Test type: Basic

A.2.3.1.6 Test Case for LocalizedContinuousString Datatype

- a) Test identifier: LocalizedContinuousString
- b) Test purpose: Verify the conformance of the product with the LocalizedContinuousString datatype. **(Mandatory)**
- c) Test method: (1) Inspect the product in order to determine that it uses values in the range of the required datatype `rdf:PlainLiteral` for properties specified with the value type LocalizedContinuousString in the NEO Standard and associated NEO content. (2) In order to satisfy the requirements for a LocalizedCharacterString in the NEO information model, a value must include both a character string and a language tag. (3) The string portion of the value must not contain any space characters (unless those are encoded using '%20').
- d) Reference: NEO Standard (Section 5.2.8.4; Section 5.4.4.2).
- e) Test type: Basic

A.2.3.1.7 Test Case for IANALanguageSubtag Datatype

- a) Test identifier: IANALanguageSubtag
- b) Test purpose: Verify the conformance of the product with the IANALanguageSubtag datatype. **(Mandatory)**

- c) Test method: (1) Inspect the product to determine that it uses values in the range of the required language datatype for properties specified with the value types `LocalizedString` and `LocalizedContinuousString` in the NEO Standard and associated NEO content. (2) In order to satisfy the requirements for an IANA LanguageSubtag in the NEO information model, a value must belong to the set of two-character, lowercase values specified in BCP 47).
- d) Reference: NEO Standard (Section 5.2.8.3; Section 5.4.4.2).

Test type: Basic

A.2.3.1.8 Test Case for Real Datatype

- a) Test identifier: Real
- b) Test purpose: Verify the conformance of the product with the Real datatype. **(Mandatory)**
- c) Test method: Inspect the product to determine that it uses values in the range of the `owl:real` datatype for properties specified with the value type Real in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.9; Section 5.4.4.2).

Test type: Basic

A.2.3.1.9 Test Case for Decimal Datatype

- a) Test identifier: Decimal
- b) Test purpose: Verify the conformance of the product with the Decimal datatype. **(Mandatory)**
- c) Test method: Inspect the product to determine that it uses values in the range of the `xsd:decimal` datatype for properties specified with the value type Decimal in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.10; Section 5.4.4.2).

Test type: Basic

A.2.3.1.10 Test Case for Integer Datatypes

- a) Test identifier: Integers
- b) Test purpose: Verify the conformance of the product with the Integer datatype. **(Mandatory)**
- c) Test method: Inspect the product to determine that it uses values in the range of the `xsd:integer` datatype, or (when specified) `xsd:nonNegativeInteger` for properties specified with the values type Integer (or, when specified, `NonNegativeInteger`) in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.11; Section 5.4.4.2).

Test type: Basic

A.2.3.2 Test Module for Measure Datatypes

- a) Test identifier: MeasureDatatype
- b) Test purpose: Verify the conformance of the product with the MeasureDatatype datatype. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it uses the correct encoding for each MeasureDatatype, as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.13; Section 5.4.4.3).
- e) Test type: Basic

A.2.3.2.1 Test Case for Measure Value

- a) Test identifier: MeasureValue
- b) Test purpose: Verify the conformance of the product with the MeasureValue property. **(Mandatory)**

- c) Test method: Inspect the product in order to determine that it uses the correct encoding (including range declaration) for the MeasureValue property of each Measure Datatype as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.13; Section 5.4.4.3).
- e) Test type: Basic

A.2.3.2.2 Test Case for Measure Unit

- a) Test identifier: MeasureUnit
- b) Test purpose: Verify the conformance of the product with the MeasureUnit property. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it uses the correct encoding for the MeasureValue property (including specification of a UnitOfMeasure from the allowable values) for each Measure Datatype as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.13; Section 5.4.4.3).
- e) Test type: Basic

A.2.3.3 Test Module for Enumerated Types

- a) Test identifier: EnumeratedTypes
- b) Test purpose: Verify the conformance of the product with the EnumeratedType datatypes and ListedValues. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it uses the correct encoding of EnumeratedType components and their ListedValues as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Sections 5.2.8.15 and 5.2.8.16; Section 5.4.4.4).
- e) Test type: Basic

A.2.3.3.1 Test Case for Enumeration

- a) Test identifier: Enumeration
- b) Test purpose: Verify the conformance of the product with the Enumeration datatype. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it uses the correct encoding of Enumerations as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.15; Section 5.4.4.4).
- e) Test type: Basic

A.2.3.3.2 Test Case for Codelist

- a) Test identifier: Codelist
- b) Test purpose: Verify the conformance of the product with the Codelist datatype. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it uses the correct encoding of Codelists as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.16; Section 5.4.4.4).
- e) Test type: Basic

A.2.3.3.3 Test Case for Listed Value

- a) Test identifier: ListedValue
- b) Test purpose: Verify the conformance of the product with the encoding of ListedValues for enumerated types. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it uses the correct encoding of ListedValues as specified in the NEO Standard and associated NEO content.

- d) Reference: NEO Standard (Table 4; Section 5.2.8.17; Section 5.4.4.4); NEO content baseline
- e) Test type: Basic

A.2.3.4 Test Module for Complex Datatypes

- a) Test identifier: ComplexDatatypes
- b) Test purpose: Verify the conformance of the product with the correct encoding of complex datatypes, including the special complex datatypes in DatatypeUnion and DatatypeMeta. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that each ComplexDatatype component is correctly encoded as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Sections 5.2.8.18, 5.2.8.19, and 5.2.8.20; Section 5.4.4.5).
- e) Test type: Basic

A.2.3.4.1 Test Case for Complex Datatype

- a) Test identifier: ComplexDatatype
- b) Test purpose: Verify the conformance of the product with the correct encoding of each ComplexDatatype datatype not included in Datatype Union or DatatypeMeta. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that it encodes complex datatypes (not included in DatatypeUnion or DatatypeMeta) as specified in the NEO Standard and associated NEO content.
- d) Reference: NEO Standard (Section 5.2.8.18; Section 5.4.4.5).
- e) Test type: Basic

A.2.3.4.2 Test Case for Datatype Union

- a) Test identifier: DatatypeUnion
- b) Test purpose: Verify the conformance of the product with the correct encoding of each DatatypeUnion datatype. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that each DatatypeUnion datatype is encoded as an `owl:Class` with a set of two or more alternative properties that are not evaluated together.
- d) Reference: NEO Standard (Section 5.2.8.19; Section 5.4.4.5).
- e) Test type: Basic

A.2.3.4.3 Test Case for Datatype Meta

- a) Test identifier: DatatypeMeta
- b) Test purpose: Verify the conformance of the product with the correct encoding of each DatatypeMeta datatype. **(Mandatory)**
- c) Test method: Inspect the product in order to determine that each DatatypeMeta datatype is encoded as an `owl:Class` with two or more properties, providing a primary data value and metadata about an evaluated property.
- d) Reference: NEO Standard (Section 5.2.8.20; Section 5.4.4.5).
- e) Test type: Basic

Annex B – ICS Pro Forma (Normative)

B.1 Introduction

An Implementation Conformance Statement (ICS) is a statement made by the supplier of an implementation or system that is claimed to conform to a given standard (or a set of standards), in which it is declared which capabilities have been implemented in the product in conformance with the standard. This is especially important when there are options that may be implemented (or not), so that a tester may evaluate the conformance of an implementation against the relevant requirements.

B.2 ICS Pro Forma for the NEO

An ICS pro forma provides a uniform means for the implementer to declare the mandatory, conditional, and optional provisions of the standard that were implemented. The NEO ICS Pro Forma shall be used by the supplier or sponsor of an implementation as a framework to document the standards-conformant capabilities of the implementation of this standard. The NEO ICS Pro Forma is on the following page.

The ICS Pro Forma shall provide the following information:

- The **Implementation Under Test (IUT)** provides the name of the realization of a specification that is the focus of the test.
- The **Test Sponsor** information includes the name, organization, and contact information for the person or organization that is submitting the implementation for test.
- The **Date of Initial ICS Completion** is the date on which the Test Sponsor submitted the completed Implementation Conformance Statement.
- The **Conformance Class** designates the set of conformance requirements pertinent to the test. The NEO Standard, Ed. 1.0 (with associated NEO content) has a single conformance class (“A”).
- The **NEO Content Baseline** identifies the version number of the NEO content to which the IUT claims conformance.
- The **Supported Encoding(s)** identifies which official NEO content encoding(s) are used by the IUT, which shall be either the RDF/XML encoding or the N-Triples encoding, or both.
- The **Test Point(s)** information specifies where the test is to be applied (e.g., at input or output from the implementation, or to static content).
- The **Test Organization** information includes the name of the organization, the POC, and contact information for the organization that is performing the conformance test.
- The **Date of Test Completion** is the date on which the Test Organization completed the conformance testing, including results returned to the Test Sponsor.

NEO – Implementation Conformance Statement (ICS)

Column Key: B = Baseline NEO S = Subset Obligation I = Implemented P/F = Pass/Fail

Column Values: M = Mandatory O = Optional C = Conditional

Implementation Under Test:

Date of Initial ICS Completion:

NEO Content Baseline (Version #):

Test Point(s):

Date of Test Completion:

Test Sponsor:

Conformance Class: A

Supported Encodings (RDF/XML and/or N-Triples):

Test Organization:

Characteristic	Parameter	Obligation			
		B	S	I	P/F
<p>General Capabilities.</p> <p>The NEO is an OWL 2 ontology of domain concepts intended for use in the NSG to consistently and accurately represent elements of shared GEOINT in data resources and applications. NEO content is encoded in RDF/XML (mandatory) and N-Triples (optional).</p> <p>Those parameters shown on the right as 'implemented' provide an indication of the capabilities enabled by the uses of NEO content produced by the implementation under test.</p> <p>These parameters are informational only; the concept of pass/fail is not applicable for this characteristic.</p>	NEO content is used for the representation (e.g., categorization) of data instances.	O			
	NEO content is used for indexing data resources.	O			
	NEO content is used to provide semantics for Linked Data.	O			
	NEO content is used to explore data resources and navigate Linked Data.	O			
	NEO content provides terms and definitions for an application that performs semantic search by leveraging concepts used to describe data resources.	O			
	NEO content is used for constraint checking of data resources.	O			
	NEO content is used by Web services that locate and/or share data resources.	O			
	NEO content is used in the mapping or integration of domain models (e.g., application schemas, taxonomies, ontologies).	O			
	NEO content is used as a reference ontology for instance-level data integration.	O			
	NEO content is used for unified querying over heterogeneous data.	O			
	NEO content is used to support inferencing over data resources to conclude implicit information from asserted information.	O			
	Other (Describe):	O			
	Other (Describe):	O			

NEO – Implementation Conformance Statement (ICS)

Column Key: **B** = Baseline NEO **S** = Subset Obligation **I** = Implemented **P/F** = Pass/Fail

Column Values: **M** = Mandatory **O** = Optional **C** = Conditional

Implementation Under Test:
Date of Initial ICS Completion:
NEO Content Baseline (Version #):
Test Point(s):
Date of Test Completion:

Test Sponsor:
Conformance Class: A
Supported Encodings (RDF/XML and/or N-Triples):
Test Organization:

Characteristic	Parameter	Obligation			
		B	S	I	P/F
<p><u>Conformance Class.</u> The Abstract Test Suite (ATS) for the NEO Standard, Edition 1.0, and associated NEO content is a compendium of abstract test cases that provide a basis for verifying the structure and content of NEO encodings. One conformance class is defined.</p>	Conformance Class A NEO content utilized or produced by this implementation conforms to the complete NEO content. An implementation must satisfy all tests in the ATS (NEO Standard, Annex A) to be conformant.	M			
	Product uses NEO content (choose at least one): ___ Via active IRI-based Web links (using the REST API component of the NSG Standards Registry). ___ From locally installed copies of the officially published technical artifacts.	M			
	The product implements the NEO structural elements including all ontology dependencies, elements with identity, generalization hierarchy, disjoint-classes axioms, and property declarations.	M			
	The product implements the documentation properties for the NEO (including both neo-ent and neo-enum) and the ontology components (Entity Classes and Properties).	M			
	The product implements the NEO datatypes and encodings for Primitive Datatypes, MeasureDatatype, EnumeratedTypes, and ComplexDatatypes.	M			
	Product represents the ontology components in separate resource files.	O			
	Authorized access to the system(s) where the product is installed is needed to test the product. ___ Explanation for how to acquire authorized access is attached to this ICS.	O			

Annex C – NEOX Utility Ontology for NSG Enterprise Ontology (Normative)

C.1 Introduction

This annex contains the specification for the NSG Enterprise Ontology Auxiliary Ontology (NEOX), a utility ontology for defining additional concepts needed for use with NEO content. The ontology is represented in OWL 2 (RDF/XML encoding).

C.2 IRIs

IRIs for official content baselines of the NEOX ontology are versioned, because the content of the NEOX may change with evolving requirements. The form of the versioned IRI is:

- IRI for NEOX (versioned): <http://api.nsgreg.nga.mil/ontology/neoX/1.0>

For convenience (where supported), this non-versioned IRI retrieves the latest version:

- IRI for NEOX (non-versioned): <http://api.nsgreg.nga.mil/ontology/neoX>

IRIs for NEOX component concepts are formed by concatenating the ontology IRI with the slash (“/”) delimiter, followed by the concept designation specified in the next section.

The namespace abbreviation for NEOX is 'neoX'.

C.3 Concepts

The specification of NEOX adheres to the information modeling concepts and encodings defined in the NEO Standard (Sections 5.2 and 5.4). Concept(s) defined in the NEOX ontology are presented in the table below, together with their sources.

Table 24 – Concept(s) in the NEOX Ontology

Ref #	Concept Designation	OWL Construct	Concept Definition	Source
1	valuesComplete	owl:DatatypeProperty	Definition: An indicator as to whether the set of listed values in an enumerated type is closed or not, with TRUE meaning 'closed' (<i>i.e.</i> , complete) and FALSE meaning 'open' (<i>i.e.</i> , not complete). Description: A set of listed values that is not complete may be extensible following specified guidelines.	Based on ISO 19103:2015, 6.5.1 Enumerations and codelists – General rules

C.4 Publication of NEOX

The NEOX utility ontology in RDF/XML (corresponding to the content of the 'neoX' namespace) is available through the REST API component of the NSG Standards Registry. This ontology is versioned for official use. Use of the non-versioned URL (where supported) retrieves the latest version.

URL for the NSG Enterprise Ontology Auxiliary Ontology (NEOX):

- URL for NEOX (versioned): <http://api.nsgreg.nga.mil/ontology/neoX/1.0>
- URL for NEOX (non-versioned): <http://api.nsgreg.nga.mil/ontology/neoX>

Individual terms may be retrieved through the REST API.

Annex D – Inspecting NEO Content (Informative)

D.1 Introduction

The NEO content is contained in two OWL ontologies and may be viewed and edited in ontology tools that load OWL encoded in the mandatory RDF/XML syntax or the optional N-Triples format. Protégé is a widely used, free, open-source ontology editor that may be used for this purpose.⁵⁰

This Annex illustrates the use of the open-source ontology viewer and editor Protégé for visualization and inspection of NEO content.

D.2 NEO Content Inspection using the Protégé Ontology Tool

D.2.1 Protégé: An Open-Source Ontology Tool for Viewing W3C OWL ontologies

The Protégé ontology editor, developed by Stanford University, is a free, open-source ontology development tool that may be used to view the NEO content. Protégé has a graphical user interface that displays the class hierarchy, with detail panes for examining specific concepts. Protégé also offers plug-ins for visualization, although the size of the NEO makes visualization of the entire ontology difficult. Protégé is available for download online at <http://protege.stanford.edu/>. Documentation about the use of the Protégé tool is available on the Protégé web site (<http://protege.stanford.edu/support.php#documentationSupport>). The Protégé wiki (http://protegewiki.stanford.edu/wiki/Main_Page) supports active user and developer communities.

NOTE: Protégé will attempt to find external resources imported by the ontology being opened. If those are not accessible from the system where Protégé is installed, Protégé will prompt the user to resolve the issue. Two options are available: (1) Click “No” to have Protégé proceed to open the ontology file without accessing the external resources; (2) locate and copy the external resources and provide them in files accessible to Protégé within the installation system (for assistance, users should consult their system administrators).

D.2.2 Viewing NEO Content using Protégé and its Plug-ins

In order to view the NEO content in Protégé, load the main NEO ontology file: ‘neo-ent.rdf’ (which imports the NEO enumerations ontology). The following screen captures show various ways of viewing information about the NEO as a whole, its component classes and properties, and their documentation, using the open-source ontology tool, Protégé.⁵¹

⁵⁰ Protégé may be downloaded online from <http://protege.stanford.edu/>.

⁵¹ Figures in this Annex were generated with Protégé 5.1.0. Different versions of Protégé may produce different graphical displays of ontology content.

Figure 15 shows the metadata information describing the NEO ontology, as presented on the Active Ontology tab of the Protégé GUI.

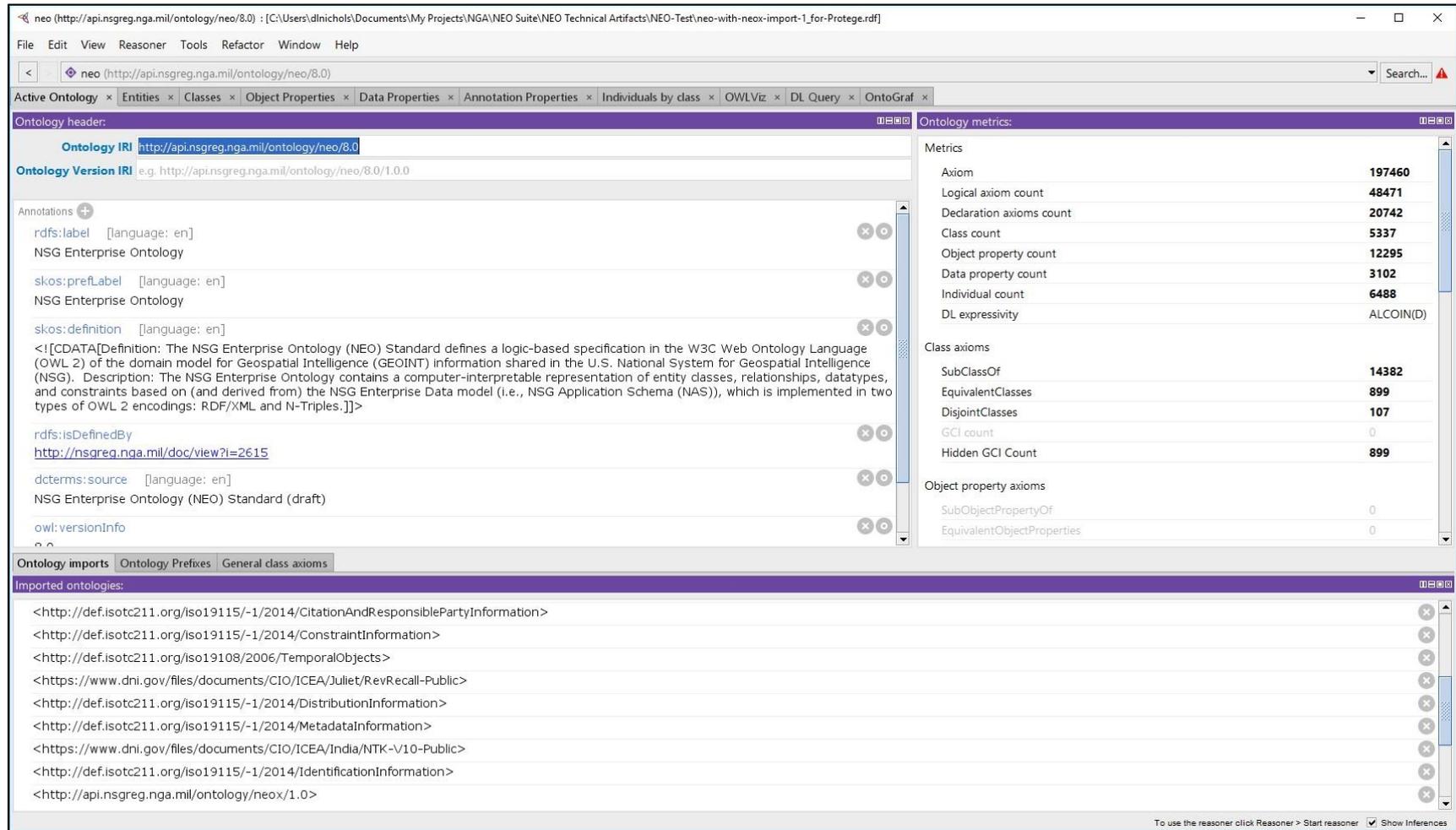


Figure 15 – NEO Described on the Active Ontology Tab of the Protégé Tool

Figure 16 shows the NEO class hierarchy as presented in the Protégé GUI (left pane), with a detailed display (upper right pane) of the documentary information for the highlighted class `FeatureEntity`, including: the preferred name of the concept, its definition, and a link to the source (*i.e.*, the NAS entry) in which the concept is defined.⁵²

The screenshot displays the Protégé GUI interface for the NEO ontology. The left pane shows a class hierarchy tree with `FeatureEntity` selected. The main pane is divided into two sections: 'Annotations: FeatureEntity' and 'Description: FeatureEntity'.

Annotations: FeatureEntity

- `rdfs:label` [language: en] FeatureEntity
- `skos:prefLabel` [language: en] Feature Entity
- `skos:definition` [language: en] <![CDATA[Definition: An abstract modelling entity that is a superclass for feature types, which are representations of temporally persistent real-world phenomena, including their geometric position and extent. Description: The concept of a 'feature' is that of the ISO 19100-series standards and as such is not constrained to any specific representation; for example, surface representation such as grids and images are allowed in addition to vector representations.]]>
- `rdfs:isDefinedBy` <http://nsgreg.nga.mil/as/view?i=100500>
- `constraint` [language: en]

Description: FeatureEntity

SubClass Of +

- `Entity`
- `FeatureEntity.locatedDeviceSurvey` max 1 owl:Thing
- `FeatureEntity.physicalObjectMetadata` max 1 owl:Thing
- `FeatureEntity.siteSignificance` max 1 owl:Thing

General class axioms +

SubClass Of (Anonymous Ancestor)

- `Entity.uniqueEntityIdentifier` exactly 1 owl:Thing
- `Entity.legalConstraints` max 1 owl:Thing
- `Entity.specifiedDomainValues` max 1 owl:Thing
- `Entity.objectIdentificationQuality` max 1 owl:Thing

Instances +

Target for Key +

Disjoint With +

- `DeviceEntity, ActorEntity, TemporalEntity, ConsumableEntity`

The bottom right corner of the window contains the text: "To use the reasoner click Reasoner > Start reasoner" and a checkbox for "Show Inferences".

⁵² The Protégé View menu (option to "Render by annotation property") may be used to set the display to the desired display string for names of classes, properties, and individuals. In Figure 16, the class names are displayed using the `rdfs:label`.

Figure 16 – Protégé NEO Hierarchy View with Definition of the Class FeatureEntity

Figure 17 shows a graphical representation of the NEO class `ActorEntity`, using the Protégé plug-in `OntoGraf` (available from the Protégé Windows > Tabs menu). The graph focus is on `ActorEntity`. Its subclasses (listed in the class hierarchy in the left panel) are displayed in the `OntoGraf` tab, together with their relationships (as arcs) to other classes. The diagram may be manipulated. Arcs may be selected or hidden using the `Arc Types` panel. Arcs in the diagram may be selected for more information. Documentation for classes may be displayed by rolling over the nodes in the graph. It is worth noting that the conventions of the `OntoGraf` diagram differ from those of UML (e.g., arrows representing the generalization relationship point towards the subclass, rather than the superclass).

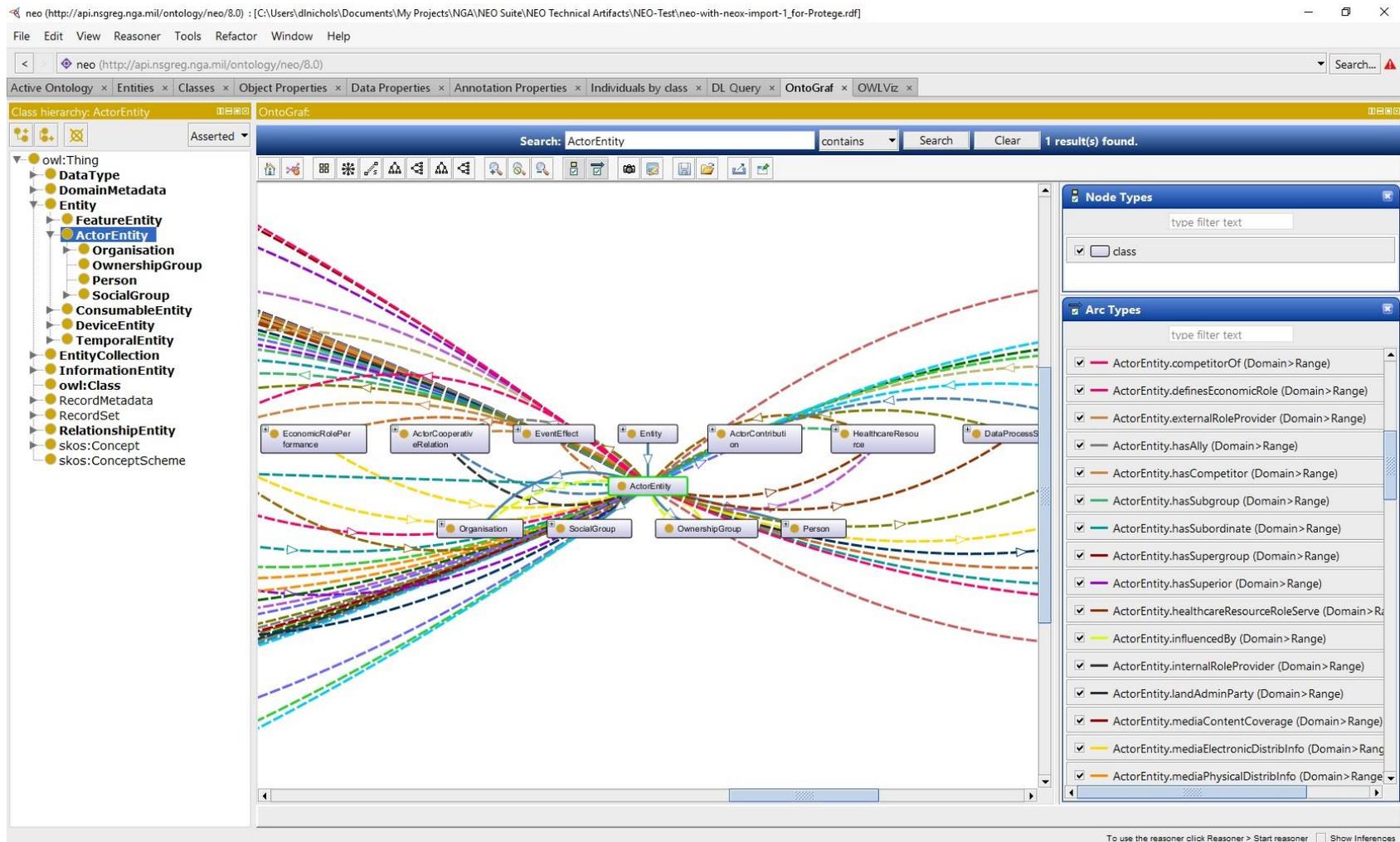


Figure 17 – OntoGraf Plug-in View of NEO ActorEntity Hierarchy with Relationships

Figure 18 shows the NEO class hierarchy (left panel) with the class *Building* selected. The top right panel displays the human-readable names and definition provided in the class annotation properties. The lower middle panel shows the superclasses (*SubclassOf*) of *Building* that represent cardinality restrictions for its properties. The lower right panel shows inherited restrictions, as well as the disjoint-classes assertions which ensure that individuals are categorized in only one entity class at this level of the hierarchy.

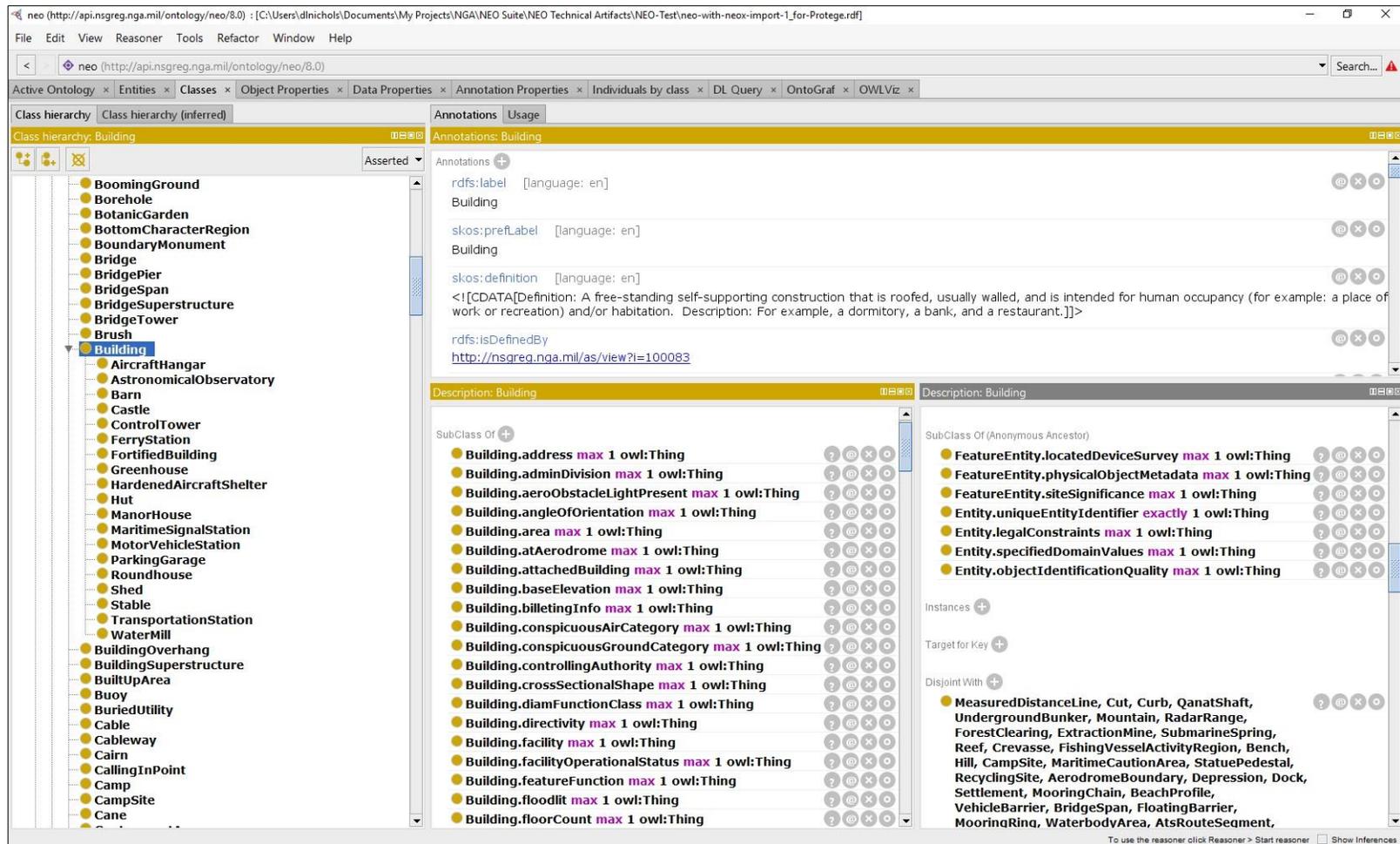


Figure 18 – Protégé Class View of NEO Building including its Subclasses, Annotations, and Property Cardinalities

In Figure 17, below, the left panel displays a section of the NEO Object Property hierarchy, showing the object properties of `Building` with the property `Building.featureFunction` selected. The upper right panel shows the annotations for the selected property, while the lower panels show the logical characteristics of the property, including its domain and range.

The screenshot shows the Protégé interface with the following components:

- Left Panel (Object Property Hierarchy):** A tree view of object properties under the `Building` class. The property `Building.featureFunction` is selected and highlighted in blue.
- Upper Right Panel (Annotations):** Displays annotations for the selected property `Building.featureFunction`. The annotations include:
 - `rdfs:label` [language: en] `featureFunction`
 - `skos:prefLabel` [language: en] `Building : Feature Function`
 - `skos:definition` [language: en] `<![CDATA[Definition: The purpose(s) of, or intended role(s) served by, the feature. Description: [None Specified]]]>`
 - `rdfs:isDefinedBy` `http://nsareg.nga.mil/as/view?!=101855`
- Lower Right Panel (Characteristics and Description):**
 - Characteristics:** A list of logical characteristics with checkboxes:
 - Functional
 - Inverse functional
 - Transitive
 - Symmetric
 - Asymmetric
 - Reflexive
 - Irreflexive
 - Description:** A list of logical relationships:
 - Equivalent To:** +
 - SubProperty Of:** +
 - Inverse Of:** +
 - Domains (Intersection):** +
 - Building**
 - Ranges (Intersection):** +
 - BuildingFeatureFunctionCodeMeta**
 - Disjoint With:** +
 - SuperProperty Of (Chain):** +

Figure 19 – Protégé View of Object Properties for Building (with `Building.featureFunction` Selected)

The range of the property `Building.featureFunction` is the class `BuildingFeatureFunctionCodeMeta`, which represents one of the NAS datatypes with metadata that are transformed into a complex datatype in NEO. NEO complex datatypes are represented by OWL classes (viewable in the Class Hierarchy, rather than in the Protégé Datatypes tab).

The Protégé Datatypes tab will not display all modeling elements that represent NEO datatypes. The Protégé display is based on the OWL Formal Specification, which defines datatypes based only on `rdf:PlainLiteral`, `rdf:XMLLiteral`, and a subset of XML Schema datatypes. As discussed in Section 5.4.4.5, NEO complex datatypes are represented in the ontology using OWL classes.⁵³

Figure 20 shows the Protégé class view of the NEO complex datatype class `BuildingFeatureFunctionCodeMeta`. This complex datatype is composed of:

- `BuildingFeatureFunctionCodeMeta.values` – a property to record the principal data value(s) (that is, one or more functions of the building);
- `BuildingFeatureFunctionCodeMeta.reason` – a property to record a reason to explain if (optionally) there is no data value; and
- Four inherited properties that may be used to record metadata about the data value (for example, the time period during which the data value is or was applicable). The metadata properties are inherited from the abstract class `DatatypeMeta`.

One or the other of these two properties will be evaluated for a data instance, either the property whose data value(s) indicate the function(s) of an individual `Building`, or the property that records the reason why there is no value.

⁵³ The representation of complex datatypes is a known problem for representing the conceptual content of a UML model using OWL 2. See J. Zedlitz and N. Luttenberger, “Data Types in UML and OWL-2”, in *SEMAPRO 2013 : The Seventh International Conference on Advances in Semantic Processing (2013)*; and J. Zedlitz and N. Luttenberger, “Transforming Between UML Conceptual Models and OWL 2 Ontologies,” in *Proceedings of the Terra Cognita Workshop on Foundations, Technologies and Applications of the Geospatial Web*, in conjunction with the 11th International Semantic Web Conference (ISWC 2012), D. Kolas, M. Perry, R. Grütter, and M. Koubarakis, Eds., 2012, pp. p. 15–26. [Online]. Available: <http://ceur-ws.org/Vol-901/paper2.pdf>

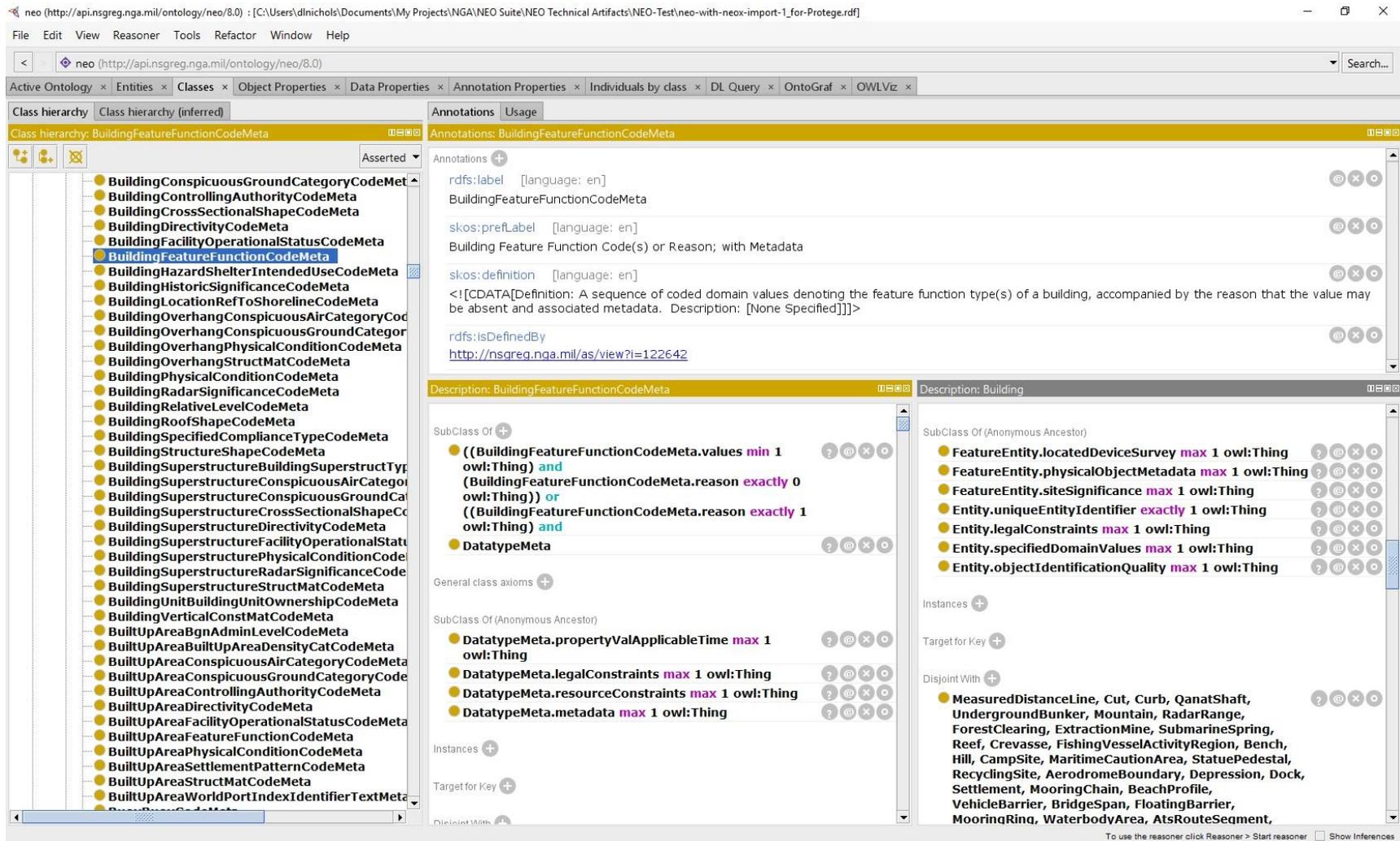


Figure 20 – NEO Complex Datatype BuildingFeatureFunctionCodeMeta

Figure 21, below, displays the Protégé view of the property `BuildingFeatureFunctionCodeMeta.values`. The property range is the class `BuildingFeatureFunction`, which represents a codelist in the Information Resources Registry of the NSG Standards Registry. The encodings of the codelist and its values (in OWL and SKOS) are available from the REST API component of the NSG Standards Registry. They may also be viewed using a web browser in the IR Registry (for example: <http://nsgreg.nga.mil/ir/view?i=100183>).

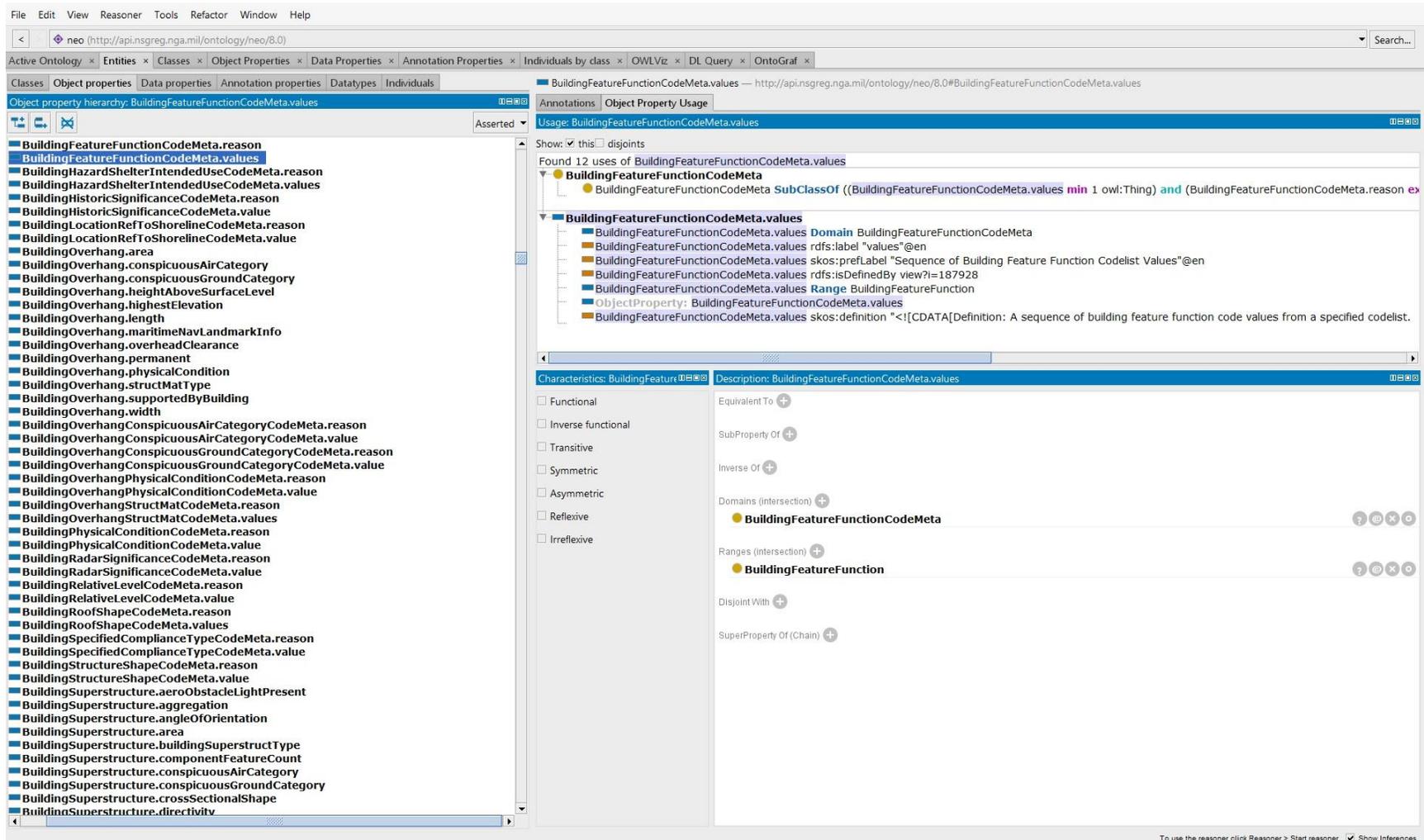


Figure 21 – Protégé View of NEO Object Property `BuildingFeatureFunctionCodeMeta.values`

The range of the property `BuildingFeatureFunction.reason` is the NEO enumeration `VoidValueReason`, whose definition and values are shown below in Figure 22. NEO enumerations are encoded as subclasses of SKOS Concept, with the listed values represented by individual SKOS Concepts. The listed values are shown in the lower right panel, below, as instances of the enumeration class `VoidValueReason`.

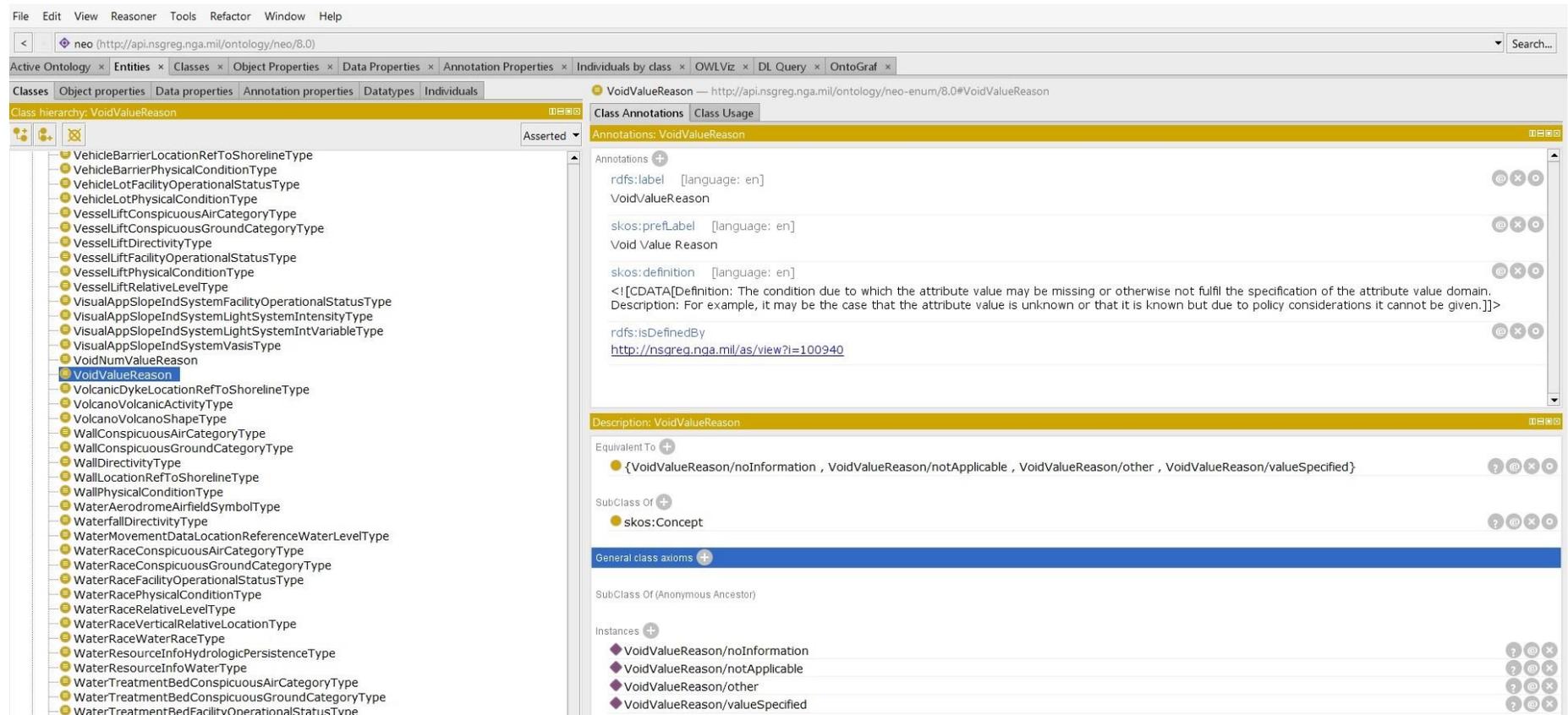


Figure 22 – Protégé Class View of the NEO Enumeration `VoidValueReason`

In addition to the various panel views and graphical displays shown above, the Protégé tool also has a search function and supports querying the ontology.

Annex E – UML Primer

(Informative)

E.1 UML Notations

The diagrams that appear in this document are presented using the Unified Modeling Language (UML) static structure diagram with the ISO Interface Definition Language basic type definitions and the UML Object Constraint Language (OCL) as the conceptual schema language. The UML notations used in this Standard are described in Figure 23.

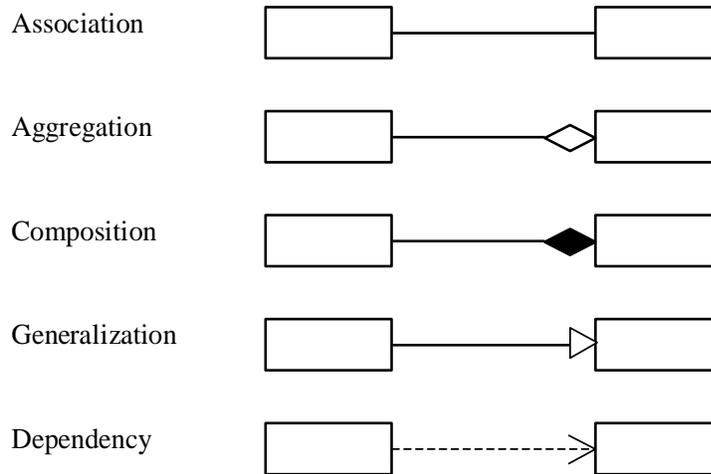


Figure 23 – UML Notation

E.2 UML Model Relationships

E.2.1 Associations

An association is used to describe a relationship between two or more classes. UML defines three different types of relationships, called association, aggregation and composition. The three types have different semantics. An ordinary association shall be used to represent a general relationship between two classes. An association may be unidirectional, *i.e.*, navigable in only one direction (indicated by an arrowhead in the direction of navigation).

The aggregation and composition associations shall be used to create part-whole relationships between two classes.

An aggregation association is a relationship between two classes in which one of the classes plays the role of container and the other plays the role of a containee.

A composition association is a strong aggregation. In a composition association, if a container object is deleted, then all of its containee objects are deleted as well. The composition association shall be used when the objects representing the parts of a container object cannot exist without the container object.

E.2.2 Navigation

Associations may be navigable in only one direction. If the direction is not specified, it is assumed to be a two-way association. If one-way associations are intended, the direction of the association can be marked by an arrow at the end of the line. Navigability means that instances participating in links at runtime (instances of an association) can be accessed efficiently from instances participating in links at the other end of the association. The precise mechanism by which such access is achieved is implementation specific. If an end is not navigable, access from the other ends may or may not be possible, and if it is, it might not be efficient.

E.2.3 Generalization

A generalization is a relationship between a superclass and the subclasses that may be substituted for it. The superclass is the generalized class, while the subclasses are specified classes.

E.2.4 Instantiation / Dependency

A dependency relationship shows that the client class depends on the supplier class/interface to provide certain services, such as:

- Client class accesses a value (constant or variable) defined in the supplier class/interface;
- Operations of the client class invoke operations of the supplier class/interface;
- Operations of the client class have signatures whose return class or arguments are instances of the supplier class/interface.

An instantiated relationship represents the act of substituting actual values for the parameters of a parameterized class or parameterized class utility to create a specialized version of the more general item.

E.2.5 Roles

If an association is navigable in a particular direction, the model shall supply a “role name” that is appropriate for the role of the target object in relation to the source object. Thus, in a two-way association, two role names will be supplied. Figure 24 represents how role names and cardinalities are expressed in UML diagrams.

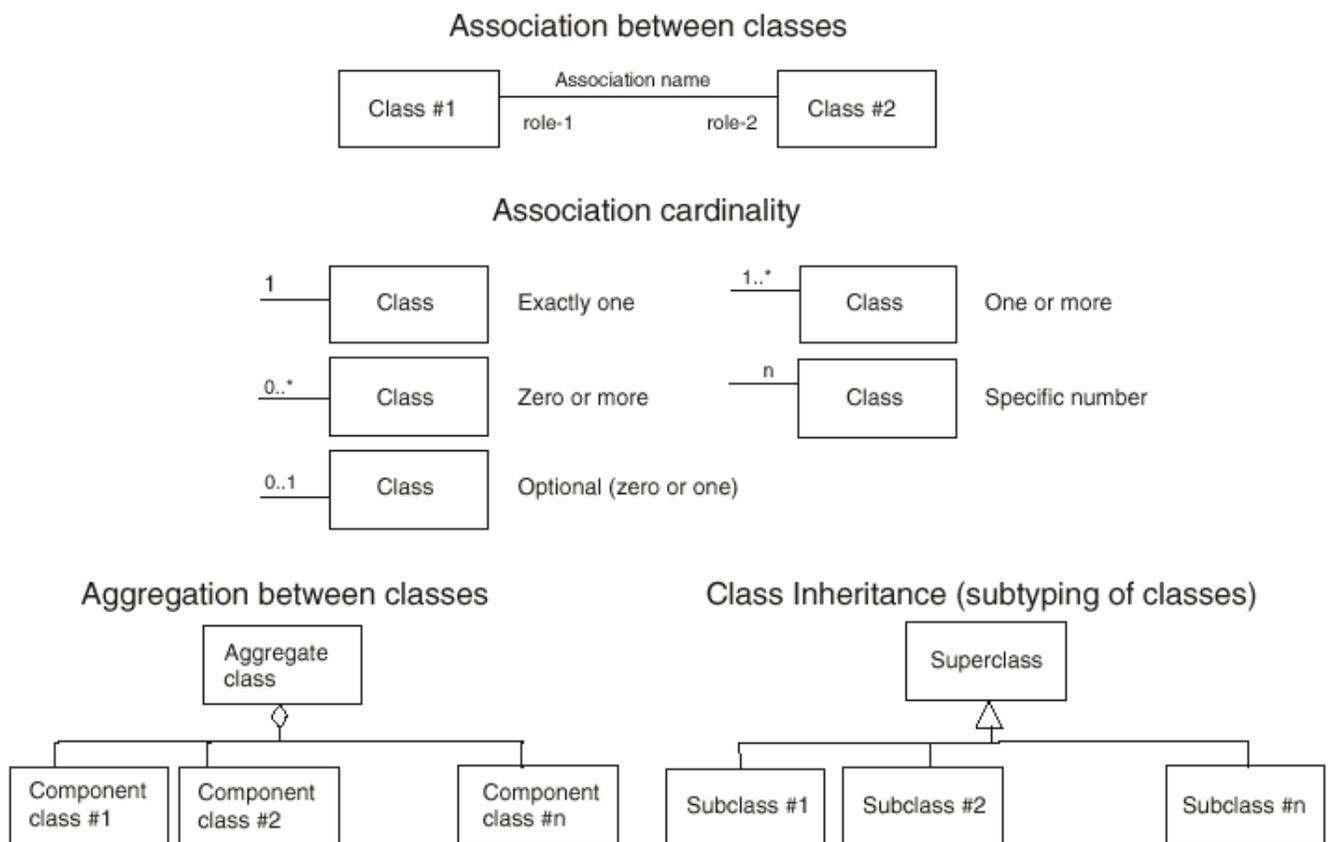


Figure 24 – UML Roles

E.3 UML Model Stereotypes

A UML stereotype is an extension mechanism for existing UML concepts. It is a model element that is used to classify (or mark) other UML elements so that they in some respect behave as if they were instances of new virtual or pseudo metamodel classes whose form is based on existing base metamodel classes. Stereotypes augment the classification mechanisms on the basis of the built-in UML metamodel class hierarchy. Below are brief descriptions of the stereotypes used in this document.

In the NSG Application Schema (NAS), the following UML stereotypes are used:

- a. <<type>> class used for specification of a domain of instances (objects), together with the operations applicable to the objects. A type may have attributes and associations.
- b. <<enumeration>> datatype whose instances form a list of named literal values. Both the enumeration name and its literal values are declared. Enumeration means a short list of well-understood potential values within a class.
- c. <<dataType>> a descriptor of a set of values that lack identity and whose operations do not have side effects. Datatypes include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string, and time. User-definable types include enumerations.
- d. <<codeList>> used to describe a more open enumeration. <<codeList>> is a flexible enumeration. Code lists are useful for expressing a long list of potential values. If the elements of the list are completely known, an enumeration should be used; if the only likely values of the elements are known, a code list should be used.
- e. <<union>> describes a selection of one of the specified types. This is useful to specify a set of alternative classes/types that can be used, without the need to create a common super-type/class.
- f. <<abstract>> class (or other classifier) that cannot be directly instantiated. The UML notation for this is to show the name in italics.