Table of Contents

Companion Specification for xAPI Vocabularies

This document is ancillary and intended to serve as a companion (as described here) to the core xAPI specification. This companion specification will provide the basic building blocks for describing xAPI vocabularies as linked datasets on the web. This draft will initially address the xAPI vocabulary requirements for Verbs and Activity Types with the expectation that semantic practices will be identified and added (as needed) for future xAPI vocabulary elements.

> An accessible, printer-ready and reader-friendly version of this companion document is also available as a gitbook here: https://adl.gitbooks.io/companion-specification-for-xapi-vocabularies/content/

Contributing to the Project

Send questions or feedback to: xapi-vocabulary@adlnet.gov

License

your contributions under the same license as the original.
- No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

- You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.
- No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Status of this Document

Owner

Name	Website	Email
xAPI Vocabulary Working Group	https://www.w3.org/community/xapivocabulary	xapi-vocabulary@adlnet.gov

Revision history

Version	Issue date	Author/editor	Description/Summary of changes
DRAFT	10/5/2015	Jason Haag	Initial Draft/Outline and primary content
DRAFT	11/18/2015	Jason Haag	Second Draft/Added information on Schema, SKOS,RDF
1.0 DRAFT	12/21/2015	Jason Haag	Third Draft/Added graphics, tables, and figures, resources, appendices, examples

Reviewed by

Version	Review Date	Name	Organization	Notes
1.0	1/15/2016	Andy Johnson	ADL	suggested cmi5 changes, etc.
1.0	1/15/2016	Tom Creighton	ADL	fixed minor typos
1.0	1/29/2016	Leslie Spotswood	ADL	fixed minor grammar mistakes & typos
1.0	1/29/2016	Dean Marvin	ADL	improved readability suggestions
1.0	2/5/2016	Russell Duhon	Saltbox	improved readability & corrections
1.0	2/24/2016	Ingo Dahn		suggested changes to process steps

Related documents

Document	Location
xAPI Vocabulary Primer	Github

License

Introduction

Designing for and developing analytics with the Experience API (xAPI) can benefit from consistent practices for semantically describing vocabulary concepts associated with the data. The requirement for a common vocabulary of Verbs and Activity Types was removed from the core xAPI specification as of the 0.95 version. Since then, the specification recommended that implementers either reuse existing vocabulary terms and identifiers or create new ones on their own. However, until now the xAPI community has lacked definitive guidelines for applying semantic practices when creating and publishing new identifiers and vocabularies.

In terms of interoperability, the core xAPI specification simply requires the data to be represented as JavaScript Object Notation (JSON) syntax, which adequately addresses structural interoperability (the ability of two or more applications or agents to exchange data). JSON is easy for humans to read and write, and easy for applications to parse and generate. This focus on structural interoperability was initially identified as the top priority of the Advanced Distributed Learning (ADL) community in order to enable the integration of learning experience data from diverse sources to any application or platform. In addition to structural interoperability, semantic data interoperability is also needed for both humans and applications to meaningfully interpret the information being exchanged. Semantic interoperability is also needed to improve vocabulary term reuse among communities of practice and to prevent duplication of terms with the same or similar meaning. In other words, without a semantic data model implementing the core xAPI specification requires more manual work to interpret, organize, aggregate, reuse, maintain, and generally do consistently useful things with the data.

In 2015, the xAPI community formed a small working group to investigate existing semantic web technologies and practices that could be leveraged for creating, publishing, and maintaining xAPI vocabularies as reusable and dereferenceable resources on the web.

This document is ancillary and intended to serve as a companion (as described here) to the core xAPI specification. This companion specification will provide the basic building blocks for describing xAPI vocabularies as linked datasets on the web. This draft will initially address the xAPI vocabulary requirements for Verbs and Activity Types with the expectation that semantic practices will be identified and added (as needed) for future xAPI vocabulary elements.

Background and Motivation

The xAPI specification provides a powerful way for applications to interoperably store and retrieve data about learning activities of users or groups of users. XAPI terms are not natural language words, but approaches to handling natural language ambiguity inform approaches to handling xAPI ambiguity. Therefore, the key driver for providing additional vocabulary guidance for xAPI is to improve consistency in the meaning of terms. When considering the semantic meaning of natural language words, there are two fundamental challenges as stated in ANSI/NISO Z39.19-2005 (2010):

1. Two or more words can be used to represent a single concept (e.g., verbs such as “finished” and “completed”).
2. Two or more words that have the same spelling can represent different concepts (e.g., verbs such as passed (to physically move past) or passed (successfully pass an examination)).

In order to address similar challenges that arise when implementing xAPI, a semantic data model and strategy for more accurately describing vocabulary terms is needed. The World Wide Web Consortium’s (W3C) Resource Description Framework (RDF) Linked Data (LD) technology stack was identified as a viable way to address this need. RDF provides a common data model and standardized syntax for identifying and describing entities (e.g., “Things”) or concepts in the world. In addition, LD compatible with JSON (JSON-LD) was released in 2014 which was naturally of interest to the xAPI community--since xAPI was already based on JSON.

By adopting LD practices for xAPI vocabularies, the xAPI can achieve a whole new level of semantic precision. Upon implementing this companion specification (and refinement of vocabulary publishing practices and processes), the xAPI community can also benefit from new capabilities generally inherent in RDF. In terms of practicality, it can immediately provide improved meaning, multilingual translation, discovery, and reuse of xAPI vocabulary terms. Conceivably, it will also open up new doors for federated search, dynamic look-up of vocabulary data within authoring applications, improved learning analytics, machine learning, and adaptive learning capabilities supported by intelligent agents and other systems.

How to Read this Document

This document provides a basic introduction regarding the application of semantic web technology to xAPI vocabularies, but also formally defines a schema expressed in Web Ontology Language (OWL). The essential parts of this schema are explained in the Vocabulary Dataset Schema. Here, the reader is presented with the set of elements that are most commonly used for representing xAPI vocabularies as linked datasets. It is expected that authors publishing vocabularies will employ most of the recommended elements presented in this section. The optional elements in the Vocabulary Dataset Schema are focused on additional metadata needs, which are likely to be used by vocabulary search interfaces and other semantic applications.

Properties and classes are **bolded**.

Code examples have a `light gray background or box`.

Vocabularies and vocabulary term identifiers are ***`bold italicized with gray background`*** when described in this document's text.

Internal hyperlinks are blue text and external links are ***bold blue italicized***.

Scope & Audience

This is a technical document written specifically for individuals and organizations implementing the xAPI. It is applicable to anyone creating new vocabulary terms (Verbs or Activity Types) for xAPI. It is also applicable to any individuals or organizations developing applications for publishing, storing or retrieving vocabulary identifiers and definitions, or any other types of services that need to query the semantic meaning of xAPI vocabulary data.

This document will provide only a brief overview of RDF. Further reading is available in the Resources Section of this document. Readers looking for additional examples should consult the xAPI Vocabulary Primer. The Primer provides real-world working examples and queries to support the implementation of this technical specification. The Primer also demonstrates semantic mapping relationships between terms, either by linking terms together or by relating them to other resources.

Conformance

This companion specification alone does not define a formal notion of conformance. However, there are three levels of obligation with regards to conformance in the core xAPI specification identified by the terms MUST, SHOULD, and MAY. Those same levels of obligation apply to this companion specification. A system or application that fails to implement a MUST requirement is non-conformant. Failing to meet a SHOULD requirement is not a violation of conformity, but goes against the recommendations of the specification. MAY indicates an option, to be decided by the developer with no consequences for conformity. Usage of these terms outside of requirement language does not designate a requirement and is avoided whenever possible. Complete definitions of MUST, SHOULD, MAY, MUST NOT, and SHOULD NOT are found in RFC 2119. Refer to Vocabulary Implementation Requirements for a list of implementation requirements for this specification.

Semantic Web Technology, Linked Data, and RDF

The “semantic web” more formally refers to a computer science field that emerged in the early 2000s, and is based on a comprehensive stack of technologies designed and recommended by the W3C. Historically, much of the information on the web has been in the form of HTML content made accessible through hyperlinks. Humans and machines both can process HTML content, but computer applications have difficulty interpreting semantic meaning from it. As a result, semantic web technology was created to solve this problem and serve as a mechanism for expressing and sharing how resources are connected and linked. A resource is commonly understood as anything that can be described and assigned an Internationalized Resource Identifier (IRI). Resources can be divided into two distinct categories (Baskauf, 2014):

1. **Information Resources**. Anything that can be transmitted electronically (i.e., digital resources). Some examples of this type of resource are html pages, images, graphics, videos, pdf files, etc.
2. **Non-information Resources** - Things that can’t be transmitted electronically. In addition, they are subdivided into one of the following categories:
    i. **Physical resource** - a tangible or material thing. Some examples of this type of resource would be a person, place, or any type of physical object or thing (e.g., an animal, a printed book, an office).
    ii. **Abstract resource** - a non-material, non-electronic thing. Some examples of this type would be a concept, a relationship, or a determination.

Linked Data (LD) is a term often used interchangeably with the semantic web. However, semantic web technologies are often inclusive of many more technology components. LD only involves applying a small subset of semantic web technologies that directly address modeling (e.g., RDF Schema [RDFS], OWL, Simple Knowledge Organization System [SKOS]), information exchange (e.g., RDF), data formats (e.g., RDFa, JSON-LD, Turtle), and leveraging the web as a platform (e.g., HTTP, IRIs). In short, LD simply refers to intentionally exposing and sharing resources on the web as RDF data. Figure 1 below provides a visual representation of the larger semantic web stack and illustrates how LD is only a subset of technologies shown here as cylinder-shaped stack.

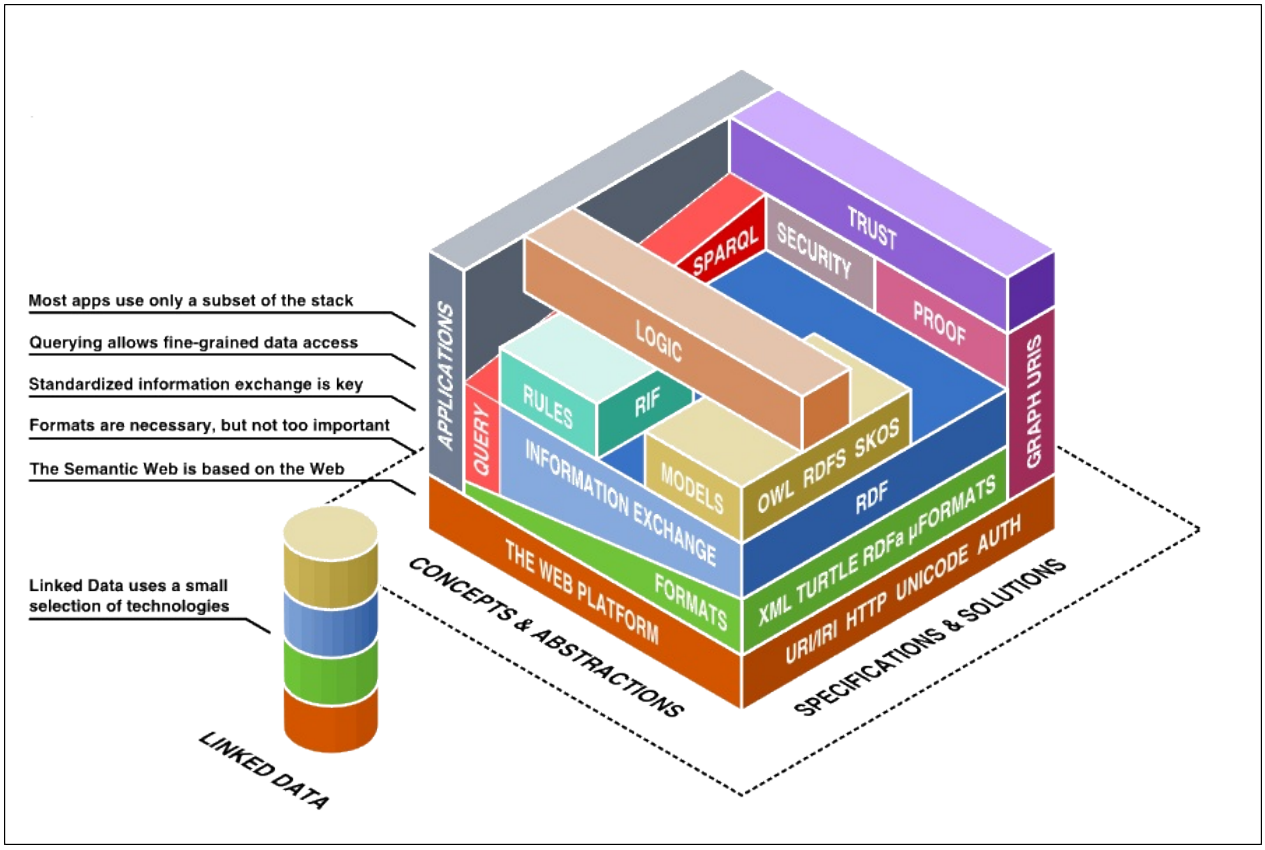

Figure 1. The semantic web technology stack. Artwork by Benjamin Nowack.

The semantic web technology stack is already being widely adopted across many diverse applications in need of a common data model and framework for publishing, sharing, reusing and aggregating data from different sources. The semantic web and LD naturally fit with and provide a foundation of knowledge representation for xAPI vocabularies. This companion specification for xAPI vocabularies will primarily focus on adopting and implementing LD. Both RDFS and OWL are used to provide a common data modeling approach or schema language for xAPI vocabularies. SKOS is an additional data model that will be leveraged. It provides specific guidelines and options for expressing relationships between vocabulary terms and associating terms with vocabularies. More details on applying SKOS are provided in Dataset Schema Design and in the xAPI Vocabulary Primer. The addition of SPARQL Query Language and Protocol (SPARQL) provides an endpoint and a protocol with a SQL-like capability for openly querying and interacting with any data represented as RDF. In other words, xAPI vocabularies can leverage this entire technology stack to aggregate and describe vocabulary resources (interrelated datasets) using RDF, accessed and dereferenced using HTTP, and queried using SPARQL.More information about SPARQL is provided in Vocabulary Search and Reuse. Some examples of SPARQL queries over xAPI vocabularies expressed as LD are also provided in the xAPI Vocabulary Primer.

RDF Data Structure

Understanding the core structure of RDF data is fundamental to describing xAPI vocabulary resources. An RDF description of a resource is a basic assertion consisting of three parts. In RDF terminology, this is known as a “triple” (*subject, predicate, object*) statement. Each part of the statement is described as follows:

1. The ***subject*** represents the resource to be described.
2. The ***predicate*** represents a property and denotes a trait being asserted about the subject (the predicate expresses a relationship between the subject and the object).
3. The ***object*** is a literal value or thing corresponding to the predicate (property).

RDF also uses IRIs to name the subject, predicate, and sometimes the object part of the triple. The object can actually be an IRI or a simple literal value. IRIs are a generalization of URIs based on RFC3986, and permit a wider range of international unicode characters. Simple literals have data types that define the range of possible values, such as strings, numbers, and dates. For example, one way to represent the notion "An xAPI verb has the English label ‘satisfied’ in RDF” is as the following triple: a subject denoting the verb (e.g., `https://w3id.org/xapi/adl/verbs/satisfied`), a predicate (property) denoting the prefered label (`http://www.w3.org/2004/02/skos/core#prefLabel`), and an object denoting the string value (e.g., “satisfied”@en”). An example RDF triple with an English language label for the ADL verb “satisfied” is further represented in the graphic below.

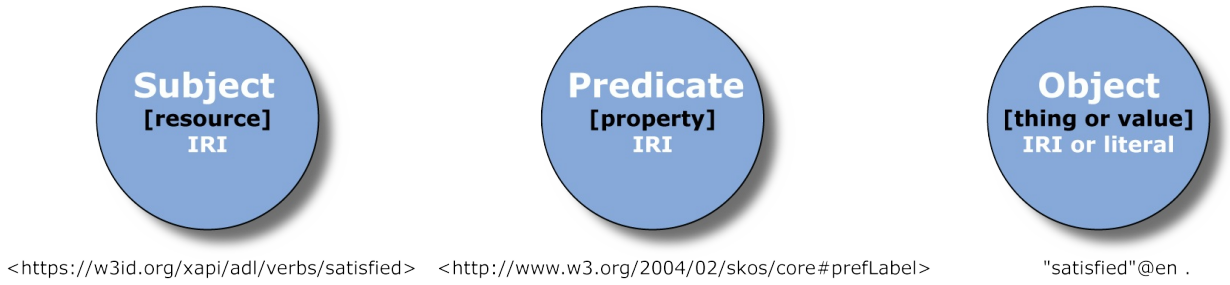

Figure 2. A Simple RDF Triple Example

IRIs are a key technology to support LD by offering a global identification solution to identify things or concepts in the world. Similarly, the core xAPI specification was intentionally designed to leverage IRIs for identifying Verbs, Activity Types, Attachments, and Extensions. When the core xAPI specification was written, it was envisioned that the IRIs used in xAPI statements could eventually map directly to vocabulary IRIs for more semantic meaning and metadata. Refer to Vocabulary Development and Publishing for an overview of IRIs and best practices for their design and persistence.

xAPI Vocabularies as Linked Datasets

In this companion document, a "dataset" is structured data describing vocabulary terms, such as xAPI Verbs. Synonymous concepts to a dataset include such things as collections, lists, or metadata record sets. A dataset may also be self-describing in that it contains information about the entire dataset itself.

Linked Datasets provide an opportunity to share common meaning and common identifiers across the xAPI community and provide consistent and reliable identifiers for a list of vocabulary terms organized as Verbs and Activity Types. A Linked Dataset is in RDF and consists of:

1. An IRI or blank node to identify each resource in the dataset.
2. An IRI for the whole dataset.
3. RDF metadata to describe the dataset and the resources in the dataset.
4. References to ontology/schema IRIs which define the concepts and relationships used within the dataset.

A basic example of the cmi5 verbs represented as an RDF dataset is available as a HTML code example in Apendix A. This example provides a representation of the dataset as RDF data embedded in HTML attributes, also known as RDFa. This is a simple approach and meets multiple requirements for xAPI vocabulary datasets to be published as both human readable (HTML) and machine readable (RDF) representations. Publishing vocabularies as RDFa is the easiest way to make vocabulary data easily discoverable by search engines and able to be queried using SPARQL and the IRI location as an endpoint. Once a dataset has been made available as RDFa, registries can also query and store the data, improving opportunities for wider vocabulary search, discovery, and reuse.

Inserting RDF attributes to HTML elements makes the data in that page able to be easily serialized in other RDF formats; however, providing JSON-LD to non RDF-aware clients requires some additional constraints to be placed on the JSON-LD document. If a dataset is also made available in other serializations such as JSON-LD then other applications (e.g., LRS, LMS, Authoring Tools) can also dynamically retrieve the complete vocabulary dataset and meaning of each of the terms. This is possible through client-server content negotiation. More information on using content negotiation to provide both HTML and JSON-LD is provided in Vocabulary Development and Publishing. A more advanced example of a vocabulary provided as a JSON-LD dataset is available in the xAPI Vocabulary Primer.

Relationship Between Vocabularies, Profiles, and Recipes

While this specification directly addresses profiles and vocabularies, it is also important to understand the distinct differences and relationships with other terminology often used interchangeably in the xAPI community. This section attempts to provide an opportunity to explain and clarify the specific terminology concepts often used in the xAPI community. Figure 3 (following the terminology descriptions below) further illustrates the characteristics of and relationship between vocabularies, profiles, and recipes. The conceptual diagram below is based on current practices. As a preliminary diagram it may be updated with feedback from the community to improve clarity.

1. **Community of Practice (CoP):** A COP is a group of practitioners connected by a common cause, role or purpose, which operates in a common modality. CoPs are focused on implementing xAPI within a specific knowledge domain or use case. CoPs, or independent developers, can create domain-specific vocabularies, profiles, and recipes. These practices usually involve work around defining use cases and curating the various vocabulary terms, synonyms, and other related metadata that might be preferred within a CoP. They canalso reuse existing vocabularies, profiles, and recipes already published by other CoPs or participants of the xAPI community.

2. **Profile:** A profile is a specific set of rules and documentation for implementing xAPI in a particular context. Profiles generally provide a particular vocabulary of terms, some created specifically for the profile, and some are referenced from other vocabularies. Sometimes a profile might provide multiple vocabularies for different situations, and sometimes someone might curate a vocabulary from multiple sources without creating a profile.

   An example of a project profile is the SCORM profile, which is designed for the traditional self-paced/single learner, online learning use case. An example statement of specifying a profile id in the statement context's category property is available here: https://github.com/adlnet/xAPI-SCORM-Profile.

   In addition to project profiles (e.g., cmi5, SCORM, IEEE ADB), more granular types of activity profiles can be created to represent specific implementations of learning activities. For example, activity profiles (e.g., assessment or video profiles) can be used as a stand-alone profile or within a broader project profile.

3. **Vocabulary:** A vocabulary is a list or collection of the terms that are used by a COP for labeling or categorizing information in a particular domain. The use of a vocabulary ensures that everyone is using the same word to mean the same thing. Vocabularies in xAPI mayuse a single list (dataset) or multiple lists (datasets) of specific terms carefully selected for use. Vocabulary datasets should be curated and organized accordingly into groups of Verbs and/or Activity Types. CoPs can publish new vocabulary datasets or reference terms from existing vocabulary sources. Each of the terms in a vocabulary dataset must have or reference a unique IRI. For example, the verb “satisfied” is identified by the IRI, `https://w3id.org/xapi/adl/verbs/satisfied`. Each vocabulary dataset must also have a unique IRI and usually follows an intelligible design pattern consistent with the IRI path of the vocabulary terms it contains (e.g., `https://w3id.org/xapi/adl`). For more details on IRI design guidelines see Generating and Maintaining IRIs

4. **Recipe:** Recipes refer to a way of expressing how a common type of learning activity could be syntactically represented in the form of a sample xAPI Statement. A recipe is intended to be reusable by other developers or CoPs. It can specify the statement structure, objects and properties, the required values associated with each object and property, and the object identifiers. Recipes are not required to represented as RDF, but may include human readable examples and descriptions at the IRI location. An example of a Recipe is available here: http://xapi.trainingevidencesystems.com/recipes/attendance/0_0_1/#simple.

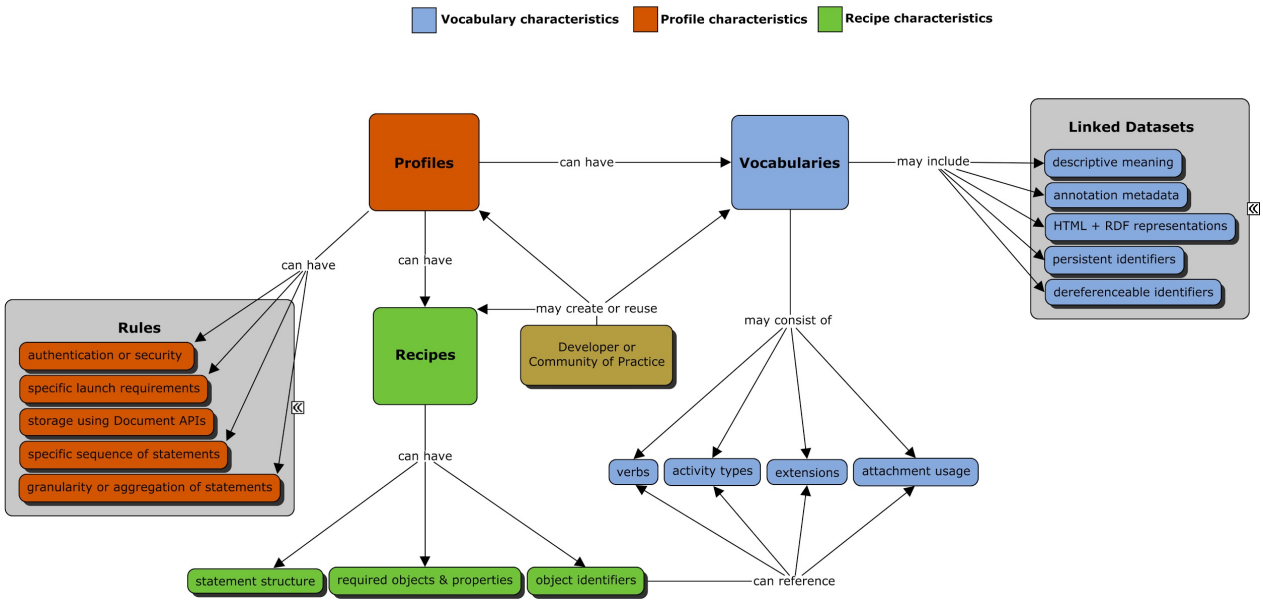

Figure 3. Relationship between vocabularies, profiles, and recipes.

Vocabulary Dataset Schema

The previous sections described the characteristics of vocabularies and the notion of expressing vocabulary data as an organized dataset. This section formally describes the data schema for xAPI vocabularies expressed as RDF. A schema is a common approach for how to define classes and attributes that should used to describe things. The term "RDF vocabulary" is sometimes used as an umbrella term in the LD community instead of schema. However, to avoid ambiguity with the general (and often overlapping) use of the word “vocabulary” in the LD community, the term “schema” will be used in this specification to best convey how Verb and Activity Type datasets are modeled and described.

It is common practice for schemas to incorporate and reuse classes and properties from pre-existing ontologies and schemas, where stable and appropriate ones could be leveraged. RDFS, OWL, and SKOS were primarily used to model the xAPI Vocabulary Dataset Schema. The recommended and optional schema elements are detailed in the tables below following Figure 4.

A list of the pre-existing ontologies and schemas used are also provided for convenience, while complete class and property descriptions are accessible from the namespace IRI column in the table. An IRI is commonly used for each of the classes and properties so that they may be applied and validated by developers and consumers of RDF data. The classes and properties of the vocabulary dataset schema containing an “**xapi:**” prefix exclusively belong to the xAPI namespace.

Namespace

The term “namespace” is used here to imply the IRI where the elements in a named ontology or schema are defined and resolved. The namespace IRI reserved for xAPI is `https://w3id.org/xapi/ontology#` . This reserved namespace IRI initially addresses xAPI vocabularies, but may be further expanded in the future to define a more complete ontological data model representation of other xAPI objects and properties as well.

Both namespaces and prefixes are practices carried over from XML schema design principles. Namespace prefixes provide a shorthand way of abbreviating long namespace IRIs, making the classes and properties that belong to a schema or ontology more readable and compact. For example, Verb defined as a Class in the xAPI Vocabulary Dataset Schema is identified by the IRI, `https://w3id.org/xapi/ontology#Verb` and may also be represented as **xapi:Verb**, if the prefix **xapi** has been declared to stand for the namespace (`https://w3id.org/xapi/ontology#`)".

The namespace prefixes and their corresponding namespace IRIs currently used in this version of the xAPI Vocabulary Dataset Schema are provided below.

Schema/Ontology Name	Prefix	Namespace IRI	Purpose for xAPI Vocabularies
Dublin Core	dcterms	http://purl.org/dc/terms/	for time/date versioning
Friend Of A Friend (FOAF)	foaf	http://xmlns.com/foaf/0.1/	for identifying the names of people or groups
Web Ontology Language (OWL)	owl	http://www.w3.org/2002/07/owl#	for ontology structure & modeling
Provenance Ontology (PROV-O)	prov	http://www.w3.org/ns/prov#	for vocabulary and term versioning an other maintenance metadata
Resource Description Framework (RDF)	rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	for describing resources an their data types
			for modeling

RDF Schema	rdfs	http://www.w3.org/2000/01/rdf-schema#	resources wit OWL classes & properties
Simple Knowledge Organization System (SKOS)	skos	http://www.w3.org/2004/02/skos/core#	for expressing relationships between concepts (e.g verbs) and concept schemes (vocabularies
Wordnet	wordnet	http://wordnet-rdf.princeton.edu/ontology#	for linking to synsets that can aid in wo sense disambiguatic and multilingual translations
Experience API (xAPI)	xapi	https://w3id.org/xapi/ontology#	to define xapi specific classes and properties
XML Schema	xsd	http://www.w3.org/2001/XMLSchema#	for when usin xsd data type such as date strings and language values

Figure 4. Table of select namespaces used for xAPI Vocabulary datasets.

Classes and Properties

RDF describes resources with classes, properties, and values. RDF Schema (RDFS), OWL, and SKOS provide a framework to model and describe the classes and properties. The following tables provide a list of both the recommended and optional classes and properties for representing xAPI vocabulary datasets as LD. It also contains a description and the functional characteristics for each class and property. In the properties table, the domain specifies for what class the property is defined (i.e., that instances of particular class can have that property). The range says what values can the property take (i.e., what type are the values it can take). Refer to the xAPI Vocabulary Primer for examples on how to apply some of the following classes and properties.

Recommended Classes

Class Name	Description	rdf:type	rdfs:subClassOf	owl:di
xapi: ActivityType	The Activity Type corresponds to the type identifier of the Activity Definition Object in an xAPI statement. When the IRI is dereferenced it can provide specific information (metadata) about the type of activity. Activity Types are concepts and should be associated with a specific activity vocabulary.	owl:Class	skos:Concept	xapi:Verb
skos:ConceptScheme	A SKOS concept scheme is an aggregation of one or more SKOS concepts. xAPI vocabularies are represented as concept schemes. For example,	owl:Class		skos:Concep

	a verb vocabulary is a concept scheme.			
xapi:Verb	Verbs correspond to the id / identifier of the Verb Object in an xAPI statement. When the IRI is dereferenced it can provide more meaning about the Verb term. Verbs are concepts and should be associated with a specific verb dataset and may be referenced by other vocabularies.	owl:Class	skos:Concept	

Figure 5. Table of recommended classes for xAPI Vocabulary datasets.

Optional Classes

Class Name	Description	rdf:type	rdfs:subClassOf	owl:disjointWith
prov:Activity	An activity is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.	owl:Class		prov:Entity

Figure 6. Table of optional classes for xAPI Vocabulary datasets.

Recommended Properties

Property Name	Description	rdf:type
xapi:closelyRelatedNaturalLanguageTerm	A property used to connect the xAPI term to a specific sense, or meaning, in natural language that is very closely related. The sense will be identified with Wordnet, a carefully constructed ontology that aids in word sense disambiguation, which has been used with many different languages.	owl:ObjectProperty
skos:definition	A property borrowed from SKOS that is used to provide a plain text definition or description for a resource.	owl:AnnotationProperty
skos:inScheme	A property borrowed from SKOS that relates a concept (for example a verb) to a concept scheme in which it is included. Verbs and Activity Types should be in a concept scheme	owl:ObjectProperty

	(vocabulary) and are connected using this property.	
skos:prefLabel	A property borrowed from SKOS that is used to provide the preferred lexical label for a resource, in a given language.	owl:AnnotationProperty
xapi:referencedBy	A property used to connects an xapi:Verb not directly maintained by the controlled vocabulary author to a vocabulary that is reusing and referencing the term.	owl:ObjectProperty
skos:scopeNote	A note that helps to clarify the meaning and/or the use of a concept.	owl:AnnotationProperty
xapi:thirdPartyLabel	A property used by people who did not create the original verb, but would like to offer an alternative label for it.	owl:AnnotationProperty

Figure 7. Table of recommended properties for xAPI Vocabulary datasets.

Optional Properties

Property Name	Description	rdf:type	rdfs:sub
skos:altLabel	Alternative label for a resource. Acronyms, abbreviations, spelling variants, and irregular plural/singular forms may be included among the alternative labels for a concept.	rdf:Property	rdfs:label
skos:broader	Relates a concept to a concept that is more general in meaning. By convention, skos:broader is only used to assert an immediate (i.e. direct) hierarchical link between two conceptual resources.	rdf:Property, owl:ObjectProperty	skos:broad
skos:broadMatch	Used to state a hierarchical mapping link between two conceptual resources in different concept schemes.	rdf:Property, owl:ObjectProperty	skos:broad skos:mapp
dcterms:created	Date of creation of the resource.	rdf:Property	dcterms:da
skos:editorialNote	A note for an editor, translator or maintainer of the vocabulary.	rdf:Property, owl:AnnotationProperty	skos:note
skos:example	Used to provide an example or contextual representation as a string. Used in combination with the prov:value property if providing a code example, such as an xAPI Statement example.	rdf:Property, owl:AnnotationProperty	skos:note
skos:historyNote	A note about the past state/use/meaning of	rdf:Property,	skos:note

	a concept.		
dcterms:modified	Date the resource was last modified.	rdf:Property	dcterms:da
foaf:name	Used to specify the name of the group or organizational activity associated with prov:wasGeneratedBy	owl:DatatypeProperty	rdfs:label
skos:narrower	Relates a concept to a concept that is more specific in meaning.	rdf:Property, owl:ObjectProperty	skos:narro
skos:narrowMatch	Used to state a hierarchical mapping link between two conceptual resources in different concept schemes.	rdf:Property, owl:ObjectProperty	skos:narro skos:mapp
skos:related	Relates a concept to a concept with which there is an associative semantic relationship.	owl:ObjectProperty, owl:SymmetricProperty	skos:sema
skos:relatedMatch	Used to state an associative mapping link between two conceptual resources in different concept schemes.	rdf:Property, owl:ObjectProperty, owl:SymmetricProperty	skos:relate skos:mapp
prov:specializationOf	An entity that is a specialization of another shares all aspects of the latter, and additionally presents more specific aspects of the same thing as the latter. In particular, the lifetime of the entity being specialized contains that of any specialization. Examples of aspects include a time period, an abstraction, and a context associated with the entity.	owl:ObjectProperty	
	Used to provide the entity that was		

prov:wasGeneratedBy	responsible for generating the resource.	owl:ObjectProperty	prov:wasIn
prov:wasRevisionOf	A revision is a derivation for which the resulting resource is a revised version of some original. The implication here is that the resulting resource contains substantial content from the original. Revision is a particular case of derivation.	owl:AnnotationProperty	

Figure 8. Table of optional properties for xAPI Vocabulary datasets.

Dataset Schema Design

The xAPI Vocabulary Dataset Schema was designed as an OWL 2 RDF-based ontology, and was produced by the xAPI Vocabulary Working Group to complement the core xAPI specification. Rather than create new classes and properties for xAPI vocabularies, the xAPI Vocabulary Dataset Schema incorporates existing ontologies and schemas that already provide capabilities for expressing relationships between the vocabulary terms and associating terms within a published vocabulary dataset. The next section will explain more specifically why SKOS and others were used.

Alignment with SKOS

A key tenet of the RDF vision is to link the vast amounts of unstructured (i.e., human-readable) information available by providing structure and semantic meaning, leading to beneficial outcomes for understanding, discovering, and sharing that information. The xAPI Vocabulary Dataset Schema is aligned with RDF and SKOS to achieve the same beneficial outcomes. SKOS, is a model for sharing or linking the meaning of related concepts. It also provides options for labels and definition translations in any language. It is commonly used for designing knowledge representations such as thesauri, classification schemes, taxonomies, subject-heading systems, or any other type of structured, controlled vocabulary. The design of the xAPI Vocabulary Dataset Schema is primarily aligned with SKOS because it provides specific guidelines and options for expressing relationships between concepts (vocabulary terms) and associating them within concept schemes (vocabularies).

xAPI Verbs and Activity Types as Concepts

In SKOS, a concept is intended to represent items in a particular knowledge domain. In the section Semantic Web Technology, Linked Data, and RDF of this document, non-information resources are defined as things that can't be transmitted electronically. One of these types of possible non-information resources is a "concept." Verbs and Activity Types fall into this category of abstract concepts. A concept can have multiple labels, but only one of these for each language can be related as a preferred label. Concepts can also be documented using different types of notes, such as definitions, scope notes, editorial notes, or history notes. SKOS further allows the establishment of links between concepts known as semantic relationships. Fully annotating these relationships in RDF is optional at this time, but will become more widely supported once the xAPI community has implemented a curation process or a tool to support it. Concepts are intended to be linked to and associated with concept schemes (i.e., vocabularies).

xAPI Vocabularies as Concept Schemes

Instances of the skos:ConceptScheme class are intended to aggregate instances of the skos:Concept class. In other words, a SKOS concept scheme is the aggregation of one or more SKOS concepts, for the purpose of this specification, a concept scheme is analogous to an xAPI vocabulary. Therefore, the **skos:ConceptScheme** class is used exclusively in the xAPI Vocabulary Dataset Schema to represent any type of xAPI vocabulary list of Verbs and/or Activity Types. SKOS does not constrain a concept to be contained within only one particular scheme. In other words, concepts (Verbs and Activity Types) can be associated with multiple concept schemes (vocabularies). This association is handled using the

skos:inScheme property. Refer to Appendix A for an example in HTML of how each of the cmi5 verbs (concept) IRIs are connected to the vocabulary (concept scheme) IRI, `https://w3id.org/xapi/cmi5`. Refer to the xAPI Vocabulary Primer for more examples.

Provenance Metadata

“Provenance” is any information about entities, activities, and people involved in producing a piece of data or thing in the world. Dublin Core is a very common source for providing provenance metadata about how things are described and organized. While iterating on examples, the xAPI Vocabulary working group determined that many of the properties from Dublin Core had overlapping similarity in SKOS and also the Provenance (PROV) Ontology. However, most of the analogous SKOS and PROV terms were deemed more appropriate. For example, **skos:prefLabel** can only be used once for each language tag with a concept, and represents the single most preferred way of labeling that concept in that language. In the interests of minimal complexity and reducing the number of external dependencies, the overlapping Dublin Core terms were excluded from the xAPI Vocabulary Dataset Schema, which helped to also eliminate the need to define additional rules outside of Dublin Core for their use in xAPI vocabularies. Dublin Core is currently used in the xAPI Vocabulary Dataset Schema only for providing specific creation and modification dates for vocabularies. Vocabulary authors are encouraged to look into the expressive constructs in PROV for all other provenance metadata. Refer to the xAPI Vocabulary Primer for examples and ideas on how to apply PROV to xAPI vocabulary datasets.

Vocabulary Development and Publishing

Developing and publishing vocabularies for xAPI require several important steps and considerations. There are many different options for the design of the IRIs and determining the location for the physical files that should be made available when the IRI is dereferenced. Vocabulary authors must also understand the purpose of supporting both human and machine readable versions--and the file types that are used for vocabulary datasets. This section of the document will explain these concepts and provide the recommended practices for xAPI vocabulary development and publishing.

IRI Design and Persistence

The concept of IRI persistence for xAPI can be explained as the desirable outcome that any HTTP IRI used as an xAPI identifier should continue to constantly identify and stably refer that resource. Since xAPI is based on a web infrastructure that is dynamic and sometimes inherently unpredictable, the community requires a better continuity strategy for creating and maintaining identifiers based on stable IRIs. The rationale behind IRI persistence is simple: imagine that a developer or CoP is reusing the Verbs defined in an existing vocabulary, and the IRIs are no longer accessible. The person or CoP reusing the Verbs would no longer know the proper definitions or meaning, intended usage, date it was last updated, and other pertinent information.

A lack of an IRI design and persistence policy, coupled with a heavy dependency on an external vocabulary, has already resulted in many xAPI Verb and Activity Type IRIs that are not accessible. This was primarily the result of adopting IRIs for Verbs and Activity Types from the JSON Activity Streams 1.0 vocabulary. These IRIs were deprecated by that community and when accessed provide no semantic meaning to xAPI implementations. A persistence strategy and guidelines are critical for stable and predictable HTTP-based IRIs. The stability of IRIs not only depends on whether they are persistent, but how CoPs prepare, design, and manage them. The purpose of this part of the specification is to provide guidance for xAPI designers or developers responsible for creating identifiers. A list of recommended persistence and design practices are provided below.

IRI Principles

This section provides a set of general design principles aimed at helping xAPI stakeholders make better decisions when creating and managing IRIs. Most of these principles were collected from IRI persistence research by the W3C's Phil Archer (2013).

1. IRIs should be stable and reliable in order to maximize the possibilities of reuse.
2. IRIs should be well-formed by following a consistent pattern for naming constructs.
3. IRIs should avoid using file extensions or technologies, and use careful naming conventions for the domain, any subdomains, and resources.
4. IRIs should avoid using query strings (e.g., `http://example.com/getId.aspx?id=1`).
5. IRIs should be designed with governance and maintenance in mind.
6. IRIs should use a dedicated and trusted service independent of the originator for provisioning and maintenance.
7. IRIs should be designed and built to be able support multiple types of formats (both human and machine readable).

Recommended IRI Design Practices

While keeping the aforementioned principles in mind, the following practices are recommended for designing all new xAPI IRIs:

1. Use a strong IRI pattern. The following pattern for an IRI designed for persistence is based on research findings from Archer (2013). In the research findings and case studies examined, the most stable IRIs followed a consistent pattern for naming constructs. The recommended pattern below has been adapted specifically for the xAPI community.

```
https:// {domain} / {resource type} / {vocabulary} / {term type} / {term}
```

For example the cmi5 verb, `https://w3id.org/xapi/adl/verbs/satisfied` follows this pattern where:

- **{domain}** is the host server where the resource .
- **{resource type}** declares the core/base type of resource (i.e., xapi).
- **{vocabulary}** specifies the vocabulary or profile that uses or maintains the vocabulary term(s) (i.e., adl).
- **{term type}** is the type of vocabulary term (i.e., verbs).
- **{term}** is the specific vocabulary term resource (i.e., satisfied).

2. Use a dedicated service when minting new xAPI IRIs. According to Yakel (2007), curation has been defined as “the active involvement of information professionals in the management, including the preservation, of digital data for future use.” Services such as W3id.org allow for the curation of persistent HTTP IRIs on the web and provide a resolution service that allows the xAPI community to manage persistent IRI identifiers that can force a redirection to a target web resource. If the target IRI changes, the persistent IRI identifier pointing to it can be easily updated. Creating an IRI identifier that can be maintained openly by multiple members of a CoP is a recommended practice as it affords a minimal number of risks and dependencies. Alternatively, designers could mint new IRIs using their own domain on their own privately owned server, but that would inherently contradict many of the aforementioned best practices for persistence.

Generating and Maintaining Vocabulary IRIs

In order to support xAPI IRI stability and curation, ADL will initially support requests for creating new persistent IRIs. If ADL were no longer operational or if governance of the xAPI changed to another organization, the w3id.org IRIs can be easily udpated via ***GitHub*** by anyone in the xAPI community or transitioned to any newly appointed governing organization. The following high-level process outlines the steps involved for minting a new IRI.

1. Verify that an existing vocabulary term IRI does not already exist by searching the xAPI community resources such as the Registry, ***http://registry.experienceapi.com*** and the xAPI Vocabulary Publishing Server (under development), ***http://xapi.vocab.pub***.
2. Create each vocabulary dataset IRI and vocabulary term IRIs using the Recommended IRI Design Practices provided in the previous section of this document.
3. Generate and publish the vocabulary as HTML/RDFa and JSON-LD using the recommended classes and properties defined in this specification. A basic example of this is provided in Appendix A. For more specific guidelines on generating and publishing the RDF, read Publishing Vocabulary Datasets as HTML/RDFa and xAPI Vocabulary Primer.
4. Use a dedicated service for creating and maintaining new IRIs. Use a service that supports IRI resolution (redirects) such as ***https://w3id.org***. Both ***http://purl.org*** and ***http://registry.experienceapi.com*** provide simple browser-based interfaces for generating IRIs. However, currently purl.org is temporarily not allowing new IRIs as it is transitioned to a new organization for governance. Currently, only the w3.org service offers a way to resolve the IRIs. This is expected to change and options will improve as this specification has been more widely adopted and implemented. For the technically savvy, the preferred way to use the https://w3id.org/xapi IRI pattern and following the aforementioned IRI design practices:
    i. ADL has forked the ***https://github.com/perma-id/w3id.org*** source code repository from GitHub.
    ii. Re-direct entries for each vocabulary dataset are already in place. Contact ADL about publishing the HTML/RDFa vocabulary on the ADL server or you can host it on your own.
    iii. If you host it on your own server, we will need to add new rules for your target server.
    iv. The w3id.org administrators will review the pull request and merge it. Once the pull request is merged, the changes go live immediately.
    v. Once the vocabulary IRIs have been published, announce and submit each new vocabulary dataset IRI to the xAPI Vocabulary Publishing Server

(***http://xapi.vocab.pub***) or each vocabulary term IRI to the Registry (e.g., currently the xAPI/TinCan Registry, ***https://registry.experienceapi.com***).

Resolving IRIs and Content Negotiation

In the core xAPI specification, IRIs may be used simply as identifiers without ever being resolved or dereferenced. However, following LD principles, vocabulary datasets and vocabulary term IRIs must be resolvable using the HTTP protocol. The primary reason for making vocabulary IRIs resolvable is to provide a reliable mechanism to obtain the actual meaning of the vocabulary terms, readable by humans or machines. Applications can use a variety of methods to generally read data, but resolving IRIs is the most common mechanism for LD. The terms "resolving" and "dereferencing" an IRI are often used interchangeably. Technically, IRI resolution is the process of determining an access mechanism and the appropriate parameters (e.g., Accept request headers in HTTP and media type). To use that access mechanism to perform an action on the IRI's resource is to "dereference" the IRI" (Baskauf, 2014). This formal definition comes from ***RFC3986***. IRI resolution may require several iterations and is often achieved through an HTTP mechanism called content negotiation.

Content negotiation is defined in the formal ***HTTP specification*** and makes it possible to serve different representations of data at the same IRI, so that user agents can resolve which representations fit their needs the best. Different client applications might have different preferences on the data format, which is indicated in the HTTP header of the request.

As previously stated non-information resources such as real-world physical objects or abstract concepts can not be transmitted electronically. Instead, the server responds to a client requesting the IRI with the HTTP response code "*303, See Other*," and the corresponding URL of a web document. This is called a 303 server-side redirect. Next, the client accepts this new target URL and retrieves a document describing the resource. For example, when accessing an xAPI vocabulary IRI (e.g., `https://w3id.org/xapi/adl`) a browser usually requests HTML to provide a human readable version while other applications looking for semantic meaning will request RDF formats such as JSON-LD. The server hosting these files selects the best match based on the request and sends the file back to the client. For example, consider the following IRIs:

- `https://w3id.org/xapi/adl` (IRI identifying the ADL Vocabulary Dataset & HTML document describing the vocabulary)
- `https://w3id.org/xapi/adl/verbs/abandoned` (IRI identifying the "abandoned" verb)

Accessing the ADL Vocabulary (`https://w3id.org/xapi/adl`) IRI in a browser will only provide a human-readable HTML representation of the dataset, but executing a HTTP GET request (e.g., `curl --header "Accept: application/ld+json" -L https://w3id.org/xapi/adl`) of

this IRI will return a response in JSON-LD. Each Verb IRI will also return the vocabulary dataset as HTML or JSON-LD through content negotiation. Content negotiation is managed on a web server and often requires administrative privileges and specialized knowledge of HTTP redirect rules. Vocabulary authors, who do not follow IRI persistence practices by leveraging a dedicated service for managing IRIs, will need to own, configure, and maintain their own server for content negotiation. Fortunately, vocabulary authors whocreate IRIs using W3C's Permanent Identifier Service, w3id.org, do not have to be concerned with content negotiation as that server is already configured to handle it. Vocabulary authors can simply generate a text-based .htaccess file and specify the target server's URL from which the negotiated files will be delivered. An .htaccess code example that uses the W3C's Apache server is available in Appendix B.

Editor's Note: The original ADL vocabulary IRI (`http://adlnet.gov/expapi`) was published before vocabulary practices were researched and established. Therefore, the original ADL vocabulary is still preserved, but for this documentation is also mirrored at `https://w3id.org/xapi/adl` to illustrate good IRI practices. When using the original ADL verbs for identifiers in actual xAPI statements, developers and CoPs should continue to use the original IRI path at `http://adlnet.gov/expapi/verbs/term` . Newly "minted" ADL vocabulary terms (e.g., cmi5 verbs) will use a persistent IRI (e.g., `http://w3id.org/xapi/adl/verbs/term`) so that IRIs can be better designed and maintained using the W3C's permanent identifier process.

Publishing Vocabulary Datasets as HTML/RDFa

There are many different ways to provide machine readable representations of xAPI vocabulary data by using RDF and content negotiation. One way to provide both human and machine readability from a single data source is by using HTML/RDFa. RDFa enables RDF data to be embedded in HTML documents, which makes it very useful for providing RDF in contexts where web publishing is limited to or focused on publishing datasets as HTML. HTML is a commonly understood format and even familiar to those people who are only moderately technical. Many search engine and social web companies are already serious about indexing RDFa content as well. In other words, RDFa is useful for not only exposing semantic data among xAPI vocabulary servers and registries, but also web search engines that can discover RDFa.

In the specific use case of xAPI, vocabulary authors should at least support generating HTML documents for human readability. Simply adding RDF inside of HTML attributes is the easiest path for empowering vocabulary authors to publish their vocabulary datasets in both a human and machine readable manner. It is expected that the xAPI Vocabulary Server, or other registry interfaces, will make this process more automated in the future through web-based forms and publishing tools.

Currently, if a new vocabulary is to be published, the developer or CoP should first generate the minimal metadata for each term using the recommended classes and properties defined in this specification. The vocabulary metadata for each vocabulary term and vocabulary (dataset) should be at least represented as both HTML and RDF by using RDFa. A basic example of this is provided in Appendix A. For more specific guidelines on manually generating the RDFa, read the steps below:

Basic Steps

The following steps are universally applicative regardless of whether the vocabulary is being hosted on a privately-owned web server or a public xAPI community resource (e.g., vocabulary server or registry).

1. Generate the list of the vocabulary terms, definitions, and other relevant metadata in a spreadsheet or some type of word processing document for collaboration and documentation purposes.
2. Determine the identifier scheme for the vocabulary and each vocabulary term by adopting stable IRI design approaches as presented in the Recommended IRI Design Practices section of this document.
3. Once the CoP or vocabulary authoring participants have agreed upon and refined the vocabulary definitions and metadata, these can be transferred into an HTML file.

Appendix A provides an example of some HTML that contains embedded RDFa. Vocabularies published as HTML/RDFa can also be visually customized using Bootstrap Themes and other presentation options or layouts as desired. For example, such a layout has been applied to the ADL Vocabulary to improve user navigation and readability, `https://w3id.org/xapi/adl`.

4. Validate the HTML and RDFa by using the ***W3C's RDFa Validator*** or other tools based on the requirements provided in ***RDFa 1.1***.
5. Upload the HTML file(s) to the target web server where the vocabulary will be hosted.
6. Follow the steps for Maintaining the IRI provided in this document, and add a rewrite rule that redirects to the IRI to the target server URL where the HTML is hosted. See Appendix B for an example of rewrite rules provided in an .htaccess file for an Apache web server.

Publishing Vocabulary Datasets as JSON-LD

In addition to providing vocabulary datasets as HTML/RDFa, vocabulary authors can also make the dataset available as JSON-LD or “JavaScript Object Notation for Linked Data.” JSON-LD is simply another way of encoding linked data using JSON, and is the preferred alternative representation of the same vocabulary dataset expressed in HTML/RDFa markup. While RDFa requires HTML, JSON-LD could also be embedded in HTML, but that would result in the author having to maintain duplicative information in the same HTML document. Instead, a JSON-LD serialization should be provided as a separate file and accessible through content negotiation. Read the Resolving IRIs and Content Negotiation section of this document for more details on content negotiation.

The importance of JSON-LD is that it may be used directly with xAPI web services. JSON-LD is extremely useful for a non-RDF consumer. In other words, JSON-LD is expected to be useful for other types of applications that wouldn’t necessarily directly rely on a SPARQL query interface to obtain vocabulary metadata. Examples of these types of applications might include xAPI authoring tools, Learning Record Store (LRS) applications, and other web applications that would benefit from consuming a JSON representation.

Currently, the vocabulary metadata for each vocabulary term and term list (dataset) should at least be represented as HTML/RDFa. It is expected that the xAPI Vocabulary Server, or other registry interfaces, will make the JSON-LD generation process more automated in the future through web-based forms and publishing tools. For now, examples of a vocabulary provided as JSON-LD is provided in the xAPI Vocabulary Primer.

Vocabulary Search and Reuse

Discoverability and search capabilities are needed to improve reuse among vocabulary authors and prevent duplication of terms with the same or similar meaning. CoPs and vocabulary authors are expected to look for existing vocabulary terms and IRIs for Verbs and Activity Types when possible (instead of building new ones from scratch). However, searching for xAPI Verbs and Activity Types have been limited and have not supported a decentralized approach until now. Embracing LD principles for publishing and exposing xAPI vocabulary datasets as RDF will allow them to be more searchable and discoverable, improving opportunities for reuse.

When IRIs resolve to RDF representations of vocabulary datasets, they can be imported into any RDF-based “triplestore,” such as the vocabulary server hosted by ADL, ***http://xapi.vocab.pub***. A triplestore, or RDF, store is a purpose-built database for the storage and retrieval of triples through semantic queries. Vocabulary datasets represented as RDF can be semantically queried using SPARQL.

SPARQL

As previously discussed in Resolving IRIs and Content Negotiation, there are already methods to retrieve remote vocabulary datasets through content negotiation and HTTP GET requests. This method is useful for applications that might require immediate access to a whole vocabulary dataset or meaning of a single term. Another, more powerful method is with the SPARQL protocol, which allows a SQL-like query statement to be sent to a service endpoint. The endpoint service exposes the data from a specific dataset IRI (aka Graph IRI) through a SPARQL query interface or it can even be returned as raw data. This method is useful for applications such as repositories or registries that might need to aggregate multiple vocabulary datasets and provide a way to browse, search, and inspect the meaning of individual terms from several unique sources. Besides querying RDF datasets from a single triplestore, SPARQL can allow for a federated search or querying of RDF datasets from multiple triplestore endpoints.

The SPARQL query language is widely used for querying RDF data and is currently supported in the xAPI Vocabulary Server, ***http://xapi.vocab.pub/sparql***. It is expected that other xAPI community resources that store xAPI vocabulary data will also provide SPARQL service endpoints upon implementing the requirements in this specification. See the Vocabulary Primer for examples of some SPARQL query statements used with xAPI vocabulary datasets and their results.

Vocabulary Implementation Requirements

The following vocabulary requirements complement those in the core xAPI specification and have been expanded to reflect the new responsibilities of creating and publishing xAPI vocabularies as linked datasets.

Requirement Number	Description	Conformance Level
1	Anyone establishing new vocabulary datasets or vocabulary terms **SHOULD** first check xAPI community resources (e.g., registries, repositories) to see if existing terms already exist to avoid duplication.	**SHOULD**
2	Anyone establishing new vocabularies and new vocabulary terms for xAPI **MUST** use HTTP IRIs so that they can be resolved/dereferenced.	**MUST**
3	Anyone establishing new vocabularies **SHOULD** use persistent and well-designed IRIs so that they can be stable for long term reliability. See the IRI Design and Persistence Section of this document for guidelines.	**SHOULD**
4	Anyone creating new vocabulary terms **MUST** provide descriptive metadata by minimally using the recommended classes and properties in this specification, and associate the terms with a vocabulary (concept scheme).	**MUST**
5	Vocabulary datasets **SHOULD** contain additional provenance metadata about the authors, version, date created, and date last modified.	**SHOULD**
6	Authors **MAY** also express relationships between terms using SKOS.	**MAY**
7	Anyone establishing a new vocabulary **MUST** publish a human-readable representation of the vocabulary dataset as HTML accessible at the IRI.	**MUST**
8	Anyone establishing a new vocabulary **MUST** publish at least one additional machine-readable representations of the vocabulary dataset in RDF (e.g., RDFa,JSON-LD,Turtle).	**MUST**
9	If multiple representations exist, vocabulary authors **SHOULD** provide a means of discovering each of the available dataset representations through content negotiation.	**SHOULD**

10	Anyone publishing a vocabulary dataset **SHOULD** announce the availability of the vocabulary and submit it to xAPI community resources (e.g., registries, repositories).	**SHOULD**
11	Repositories, registries, or any other systems that publish xAPI vocabulary datasets expressed as RDF **SHOULD** provide a public SPARQL endpoint for querying that data.	**SHOULD**
12	Since vocabularies can change, a system aggregating vocabularies **SHOULD** have a process to keep their versions up to date.	**SHOULD**

Figure 9. Table of xAPI vocabulary requirements.

References

Archer, P. (2013, June 24). Study on Persistent URIs. Retrieved December 22, 2015, from http://philarcher.org/diary/2013/uripersistence/

Baskauf, S. (2014). Beginner's guide to RDF: 1. Resources and URIs. Biodiversity Information Standards Working Group.Retrieved from https://code.google.com/p/tdwg-rdf/wiki/Beginners1URIs

Heath, T., & Bizer, C. (2011) Linked Data: Evolving the web into a global data space. In Y. Ding and P. Groth (Eds.), Synthesis lectures on the semantic web: Theory and technology (1st edition) (1-136). San Rafael, CA: Morgan & Claypool.

National Information Standards Organization (2010). Guidelines for the construction, format, and management of controlled vocabularies ANSI/NISO Z39.19-2005 (R2010). Retrieved from http://niso.org

Princeton University. (2010). WordNet. Princeton University. http://wordnet.princeton.edu

W3C: The World Wide Web Consortium. (2009). Simple Knowledge Organization System (SKOS) primer. Retrieved January 7, 2015, from http://www.w3.org/TR/skos-primer

Yakel, E. (2007) "Digital curation", OCLC Systems & Services: International digital library perspectives, Vol. 23 Iss: 4, pp.335 - 340

W3C: The World Wide Web Consortium. (2014). Best practices for publishing Linked Data. Retrieved November 9, 2015, from http://www.w3.org/TR/ld-bp/

Additional Resources

Architecture of the Web (URI Persistence): http://www.w3.org/TR/webarch/#URI-persistence

Beginner's Guide to RDF: https://code.google.com/p/tdwg-rdf/wiki/Beginners

Best Practices for Publishing Linked Data: http://www.w3.org/TR/ld-bp/

Best Practices for Publishing RDF Vocabularies: http://www.w3.org/TR/swbp-vocab-pub/

Cool URIs for the Semantic Web: http://www.w3.org/TR/cooluris/

Data on the Web Best Practices & Requirements: http://www.w3.org/TR/dwbp-ucr/

Ten Rules for Persistent URIs: http://philarcher.org/diary/2013/uripersistence/

The RDF Primer: http://www.w3.org/TR/rdf11-primer

Acknowledgements

The development of the vocabulary schema was facilitated by the Advanced Distributed Learning (ADL) Initiative and developed by the xAPI Vocabulary Working Group. ADL would like to recognize the following experts for their participation in developing use cases and contributions to the group: Pankaj Agrawal, Kirby Bell, Jessie Chuang, Adam Cooper, Ingo Dahn, Tom De Nies, Russell Duhon, Diane Hillmann, Jonathan Kevan, Jason Lewis, Elf Pavlik, Jon Phipps, Mark Spivey, and Bruno Winck.

This companion vocabulary specification was also produced by ADL, and its contents reflect extensive discussion within the Working Group as a whole. The editors would like to recognize the considerable contributions and input provided by the following people: Ingo Dahn, Andrew Downes, and Russell Duhon.

Appendices

Appendix A: HTML/RDFa Vocabulary Dataset Example

Vocabulary Dataset as RDFa. ***This example is not complete and is provided for documentation purposes only. The cmi5 verbs are actually part of the ADL vocabulary dataset.***

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Experience API (xAPI) - cmi5 Vocabulary Dataset</title>
</head>
<body>
<div xmlns="http://www.w3.org/1999/xhtml" prefix="
    dcterms: http://purl.org/dc/terms/
    foaf: http://xmlns.com/foaf/0.1/
    owl: http://www.w3.org/2002/07/owl#
    prov: http://www.w3.org/ns/prov#
    rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
    rdfs: http://www.w3.org/2000/01/rdf-schema#
    skos: http://www.w3.org/2004/02/skos/core#
    xapi: https://w3id.org/xapi/ontology#
    xsd: http://www.w3.org/2001/XMLSchema#">

  <!-- Vocabulary -->
  <div typeof="skos:ConceptScheme" about="https://w3id.org/xapi/cmi5">
    <div property="skos:prefLabel" lang="en" xml:lang="en" content="cmi5 Vocabulary">
      <h2 class="page-header">cmi5 Vocabulary</h2>
    </div>
  </div>

  <table class="table table-hover">
    <thead>
      <tr class="info">
        <th>Label</th>
        <th>Description</th>
        <th>Closely Related Term</th>
        <th>Vocabulary</th>
      </tr>
    </thead>

    <!-- Newly Minted Verb -->
    <tbody typeof="xapi:Verb" about="https://w3id.org/xapi/adl/verbs/abandoned" id="ab
andoned">
      <tr>
        <td property="skos:prefLabel" lang="en" xml:lang="en" content="abandoned">aban
doned</td>
```

```
        <td property="skos:definition" lang="en" xml:lang="en" content="Indicates the
activity provider has determined that the session was abnormally terminated either by
an actor or due to a system failure.">Indicates the activity provider has determined t
hat the session was abnormally terminated either by an actor or due to a system failur
e.</td>
        <td rel="xapi:closelyRelatedNaturalLanguageTerm" resource="http://wordnet-rdf.
princeton.edu/wn31/202080923-v"><a href="http://wordnet-rdf.princeton.edu/wn31/2020809
23-v" target="_blank">http://wordnet-rdf.princeton.edu/wn31/202080923-v</a></td>
        <td rel="skos:inScheme" resource="https://w3id.org/xapi/cmi5"><a href="https:/
/w3id.org/xapi/cmi5">https://w3id.org/xapi/cmi5</a></td>
      </tr>
    </tbody>

    <!-- Newly Minted Verb -->
    <tbody typeof="xapi:Verb" about="https://w3id.org/xapi/adl/verbs/satisfied" id="sa
tisfied">
      <tr>
        <td property="skos:prefLabel" lang="en" xml:lang="en" content="satisfied">sati
sfied</td>
        <td property="skos:definition" lang="en" xml:lang="en" content="Indicates that
 the authority or activity provider determined the actor has fulfilled the criteria of
 the object or activity.">Indicates that the authority or activity provider determined
 the actor has fulfilled the criteria of the object or activity.</td>
        <td rel="xapi:closelyRelatedNaturalLanguageTerm" resource="http://wordnet-rdf.
princeton.edu/wn31/202677669-v"><a href="http://wordnet-rdf.princeton.edu/wn31/2026776
69-v" target="_blank">http://wordnet-rdf.princeton.edu/wn31/202677669-v</a></td>
        <td rel="skos:inScheme" resource="https://w3id.org/xapi/cmi5"><a href="https:/
/w3id.org/xapi/cmi5">https://w3id.org/xapi/cmi5</a></td>
      </tr>
    </tbody>

    <!-- Newly Minted Verb -->
    <tbody typeof="xapi:Verb" about="https://w3id.org/xapi/adl/verbs/waived" id="waive
d">
      <tr>
        <td property="skos:prefLabel" lang="en" xml:lang="en" content="waived">waived<
/td>
        <td property="skos:definition" lang="en" xml:lang="en" content="Indicates that
 the learning activity requirements were met by means other than completing the activi
ty. A waived statement is used to indicate that the activity may be skipped by the act
or.">Indicates that the learning activity requirements were met by means other than co
mpleting the activity. A waived statement is used to indicate that the activity may be
 skipped by the actor.</td>
        <td rel="xapi:closelyRelatedNaturalLanguageTerm" resource="http://wordnet-rdf.
princeton.edu/wn31/202539728-v"><a href="http://wordnet-rdf.princeton.edu/wn31/2025397
28-v" target="_blank">http://wordnet-rdf.princeton.edu/wn31/202539728-v</a></td>
        <td rel="skos:inScheme" resource="https://w3id.org/xapi/cmi5"><a href="https:/
/w3id.org/xapi/cmi5">https://w3id.org/xapi/cmi5</a></td>
      </tr>
    </tbody>
  </table>
</div>
</body>
```

```
  </html>
```

Figure 10. Example of vocabulary as RDFa/HTML (index.html hosted at `http://example.com/datasets/cmi5`)

Appendix B: Content Negotiation .HTACCESS (Apache) Example

Editor's note: This shortened example is provided for documentation purposes only and only shows how this can be accomplished using an Apache web server. Other web servers could be used such as NGINX. Please refer to the specific type of documentation for other web servers on how to address content negotiation and URL rewriting.

```
Options +FollowSymLinks
RewriteEngine on

# vocabulary dataset
# --------------------------
# Rewrite rule to serve HTML content from the vocabulary IRI if requested
RewriteCond %{HTTP_ACCEPT} !application/rdf\+xml.*(text/html|application/xhtml\+xml)
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/.*
RewriteRule ^cmi5$ http://example.com/datasets/cmi5/index.html [R=303]

# Rewrite rule to serve JSON-LD content from the vocabulary URI if requested
RewriteCond %{HTTP_ACCEPT} application/ld\+json
RewriteRule ^cmi5$ http://example.com/datasets/cmi5.jsonld [R=303]

# verbs & activity types catch all IRIs
# --------------------------
# Rewrite rule to serve HTML content from the vocabulary term IRI if requested
RewriteCond %{HTTP_ACCEPT} !application/rdf\+xml.*(text/html|application/xhtml\+xml)
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/.*
RewriteRule ^cmi5/([a-z-]+)$ http://example.com/datasets/cmi5/#$1 [R=303,NE]

# Rewrite rule to serve JSON-LD content from the vocabulary term IRI if requested
RewriteCond %{HTTP_ACCEPT} application/ld\+json
RewriteRule ^cmi5/[a-z-]+$ http://example.com/datasets/cmi5.jsonld [R=303]
```

Figure 11. Example of .htaccess (assuming the identifier base pattern is https://w3id.org/xapi/cmi5)