IEEE SA
STANDARDS ASSOCIATION

IEEE Standard for Learning Technology—JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access

IEEE STANDARDS

IEEE Computer Society

Developed by the
Learning Technology Standards Committee

IEEE Std 9274.1.1™-2023

IEEE

IEEE Standard for Learning Technology—JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access

Developed by the

Learning Technology Standards Committee
of the
IEEE Computer Society

Approved 30 March 2023

IEEE SA Standards Board

Abstract: This standard is a collaborative effort to improve and standardize the 1.0.3 version Experience Application Programming Interface (xAPI) specification. This Standard describes a JavaScript Object Notation (JSON) data model format and a Representational State Transfer (RESTful) Web Service Application Programming Interface (API) for communication between Activities experienced by an individual, group, or other entity and a Learning Record Store (LRS). The LRS is a system that exposes the RESTful Web Service API for the purpose of tracking and accessing experiential data, especially in learning and human performance.

Keywords: Experience API, IEEE 9274™, IEEE 9274.1.1™, JavaScript object notation, JSON, Learning Record Provider, Learning Record Store, LRP, LRS, representational state transfer, REST, xAPI

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 979-8-8557-0031-2 STD26400
Print: ISBN 979-8-8557-0032-9 STDPD26400

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (https://standards.ieee.org/ipr/disclaimers.html), appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement IEEE Standards documents do not guarantee safety, security, health, or environmental protection, or guarantee against interference with or from other devices or networks. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon their own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus balloting process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group. Statements made by volunteers may not represent the formal position of their employer(s) or affiliation(s).

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents**.

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and subcommittees of the IEEE SA Board of Governors are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the IEEE SA myProject system.[1] An IEEE Account is needed to access the application.

Comments on standards should be submitted using the Contact Us form.[2]

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

[1]Available at: https://development.standards.ieee.org/myproject-web/public/view.html#landing.
[2]Available at: https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, neither IEEE nor its licensors waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; https://www.copyright.com/. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore or contact IEEE.[3] For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE SA Website.[4] Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore. Users are encouraged to periodically check for errata.

Patents

IEEE standards are developed in compliance with the IEEE SA Patent Policy.[5]

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has

[3]Available at: https://ieeexplore.ieee.org/browse/standards/collection/ieee.
[4]Available at: https://standards.ieee.org/standard/index.html.
[5]Available at: https://standards.ieee.org/about/sasb/patcom/materials.html.

filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at https://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IMPORTANT NOTICE

Technologies, application of technologies, and recommended procedures in various industries evolve over time. The IEEE standards development process allows participants to review developments in industries, technologies, and practices, and to determine what, if any, updates should be made to the IEEE standard. During this evolution, the technologies and recommendations in IEEE standards may be implemented in ways not foreseen during the standard's development. IEEE standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Participants

Experience API Specification documents are a product of Research and Development efforts led by the Advanced Distributed Learning Initiative, which is a DoD program under the Office of the Deputy Assistant Secretary of Defense (Force Education and Training).

At the time this IEEE standard was completed, the Experience API Base Standard Working Group had the following membership:

Jonathan Poltrack, *Chair*
Andy Johnson, *Vice Chair, Techical Editor*
Nick Washburn, *Secretary*

Pankaj Agrawal
Anthony Altieri
Avron Barr
Kirby Bell
Tiajuana Benson-Bond
Shelly Blake-Plock
Mitch Bonnett
Megan Bowe
Brenda Braitling
Keith Brawner
Jessie Chuang
Ben Clark
Jeffrey Clem
Thomas Creighton
Marco Dal Colle
Brandt Dargue
Deborah Decker
Tim Dickinson
Burt Dillenbeck
Peter Dobinson
Jonathan Dudzik
Erick Emde
Becky Goldberg
Jason Haag
Viktor Haag
Mike Hernandez
William Hoyt
Moges Kelklie
Jonathan Kevan
Daniel McCoy
Matthew McGuire
Ryan Mizusaki
Alan Mustafa
Freddie O'Connell
Scott Powers
Sam Rogers
Hamadou Saliah-Hassane
Domenic Schipani
Aaron Silvers
Charles Touron
George Vilches
Michael Weinraub
Luis Felipe Zapata Rivera

The working group would like to acknowledge the effort of the following individuals in the creation of the Experience API and Experience API community.

Dan Allen
Catherine Allman
Jonathan Archibald
Keith Bartolotta
Steve Baumgartner
Al Bejcek
Marcus Birtwhistle
Joerg Boeselt
Jeremy Brockman
Jennifer Cameron
Rob Chadwick
Rich Chetwynd
Lewis Cowper
Ingo Dahn
Mark Davis
Jhorlin De Armas
Andrew Downes
Andriy Drozdyuk
Brian K. Duck
Russell Duhon
David Ells
Paul Esch
Michael Flores
Steve Flowers
Richard Fouchaux
Walt Grata
Joe Gorup
Doug Hagy
Chaim-Leib Halbert
Dennis Hall
Chris Handorf
Luke Hickey
Thomas Ho
Lang Holloman
Nikolaus Hruska
Loic Jeannin
David N. Johnson
Eric Johnson
Patrick Kedziora
John Kleeman
Dan Kuemmel
Richard Lenz
Jason Lewis
Robert Lowe
Zach Lowry
Bill McDonald
Brian J. Miller
Kris Miller
Fiona Leteney
Tim Martin
Dave Mozealous
Tyler Mulligan
Mike Palmer
David Pate
Danny Pham
Jeff Place
Rick Raymer
Michael Roberts
Paul Roberts
Kris Rockwell

Jennifer Rogers
Mike Rustici
Chris Sawwa
Matteo Scaramuccia
Ángel Serrano
Ali Shahrazad
Ryan Smith
Roger Swetnam
Greg Tatka
Stephen Trevorrow
Chad Udell
Anto Valan
Melanie VanHorn
Yannick Warnier
Michael Wheeler
Andy Whitaker
Lou Wolford
R.J. Zaworski

The following members of the individual Standards Association balloting group voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Robert Aiello
Charles Barest
Avron Barr
Shelly Blake-Plock
Thomas Creighton
David Fuschi
James Goodell
Werner Hoelzl
William Hoyt
Andy Johnson
Piotr Karocki
Lakshman Raut
Linda Steedman
Andreas Tsouchlaris
Lisa Ward
Karl Weber
Yu Yuan
Janusz Zalewski

When the IEEE SA Standards Board approved this standard on 30 March 2023, it had the following membership:

David J. Law, *Chair*
Ted Burse, *Vice Chair*
Gary Hoffman, *Past Chair*
Konstantinos Karachalios, *Secretary*

Sara R. Biyabani
Doug Edwards
Ramy Ahmed Fathy
Guido R. Hiertz
Yousef Kimiagar
Joseph L. Koepfinger*
Thomas Koshy
John D. Kulick
Joseph S. Levy
Howard Li
Johnny Daozhuang Lin
Gui Lin
Xiaohui Liu
Kevin W. Lu
Daleep C. Mohla
Andrew Myles
Paul Nikolich
Annette D. Reilly
Robby Robson
Lei Wang
F.Keith Waters
Karl Weber
Philip B. Winston
Don Wright

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 9274.1.1-2023, IEEE Standard for Learning Technology—JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access.

Today's learning ecosystems, digital applications, content, and web-based tools take advantage of large amounts of learning data to provide learning analytics to improve curricula, to apply artificial intelligence for the purpose of making recommendations, to visualize data in ways that leverage advances in both data logistics and human-centered computing, and many other emerging use cases. Due to a lack of standardized technologies for these use cases, the vendors creating these platforms are forced to create proprietary solutions. The result is a locked-in environment where a single vendor may levy undue control over all parts of a learning ecosystem—to the detriment of learning objectives.

In response to this modernization, an Open-Source collaborative effort resulted in the creation of the Experience API (xAPI). This Standard is a collaborative effort involving that same community, partnered with the IEEE to improve upon and standardize version 1.0.3 of the xAPI. Artifacts and the latest versions of this specification can be found at https://opensource.ieee.org/xapi/xapi-base-standard-documentation.

This standard describes a JavaScript Object Notation (JSON) data model format and a Representational State Transfer (RESTful) Web Service Application Programming Interface (API) for communication between Activities experienced by an individual, group, or other entity and a Learning Record Store (LRS). The LRS is a system that exposes the RESTful Web Service API for the purpose of tracking and accessing experiential data, especially in learning and human performance.

Historical contributors

In collection of requirements for the xAPI, many people and organizations provided invaluable feedback to the SCORM, distributed learning efforts, and learning technology efforts in general. While not an exhaustive listing, the white papers gathered in 2008 by the Learning Education and Training Standards Interoperability (LETSI) group, the Rustici Software UserVoice website, one-on-one interviews and various blogs were important sources from which requirements were gathered for the xAPI specification.

ADL's role in the Experirence API (xAPI)

The Advanced Distributed Learning (ADL) Initiative took on the roles of steward and facilitator in the development of the xAPI. The xAPI is seen as one piece of the ADL Total Learning Architecture (previously the Training and Learning Architecture), which facilitates learning anytime and anywhere. ADL views the xAPI as an evolved version of Sharable Content Object Reference Model (SCORM) that can support similar use cases but can also support many of the use cases gathered by ADL and submitted by those involved in distributed learning that SCORM could not enable.

How to contribute

To contribute to this effort, please register to join the IEEE SA 9274.1.1 Working Group

Contents

IEEE Standard for Learning Technology—JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access

1. Overview

The Experience API (xAPI) is a standard that describes an interoperable means to document and communicate information about learning experiences. It specifies a structure to describe learning experiences and defines how these descriptions can be exchanged electronically.

In assessing candidates' suitability for positions or their capability for performing various tasks, there is a need to consider a wide range of formal and informal learning experiences, both on and offline. That information, more often than not, is scattered across a wide variety of sources.

Out of this decentralized environment, the Advanced Distributed Learning (ADL) Initiative created the original xAPI community and specification. The working group effort was moved to the IEEE in 2019.

xAPI assumes that:

— There is a need to be able to analyze information about learning experiences and their outcomes distributed across a wide variety of sources, platforms and technologies.

— Developing a commonly-accepted framework for gathering, storing and exchanging this information represents the best way of achieving this.

The goals of the xAPI are:

— To make it easier to understand and compare learning experiences and their outcomes recorded across a wide variety of contexts, platforms and technologies.

— To maximize interoperability of services which create, gather, store and process information about learning experiences.

— To provide a guide to those who want to build applications that conform to and implement this specification.

— To provide criteria against which conformance to this specification can be tested.

The xAPI Base Standard is an IEEE Open Source Project hosted on the IEEE Open Source Platform (it is licensed under the terms of the Apache 2.0 License and copyright the IEEE XAPI Authors see xAPI About for licensing and copyright information as well as additional information on contributing to this open source project). You can find this document and other open source resources related to the xAPI Base Standard at https://xapi.ieee-saopen.org.

Additional xAPI Base Standard resources are available as described below:

— xAPI Base Standard documentation

— xAPI Base Standard markdown

— xAPI Examples (See also Annex A)

1.1 Scope

This standard describes a JavaScript Object Notation (JSON) data model format and a Representational State Transfer (RESTful) Web Service Application Programming Interface (API) for communication between Activities experienced by an individual, group, or other entity and a Learning Record Store (LRS). The LRS is a system that exposes the xAPI RESTful Web Service API for the purpose of tracking and accessing experiential data, especially in learning and human performance.

1.2 Purpose

The purpose of this standard is to provide an interoperable means to store and retrieve learning experience data as required by modern, data-intensive learning technologies.

1.3 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).[6,7]

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

[6]The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.
[7]The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they *shall* be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

FIPS PUB 180-2, Secure Hash Signature Standard (SHA2).[8]

IEEE Std 754™, IEEE Standard for Floating-Point Arithmetic.[9,10]

IETF RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.[11]

IETF RFC 2616, Hypertext Transfer Protocol – HTTP/1.1.

IETF RFC 3629, UTF-8, a transformation format of ISO 10646.

IETF RFC 3987, Internationalized Resource Identifiers (IRIs).

IETF RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace (IRIs).

IETF RFC 5646, Tags for Identifying Languages.

IETF RFC 7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.

IETF RFC 7515, JSON Web Signature (JWS).

IETF RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format.

ISO 639, Code for the Representation of Names of Languages.[12]

ISO 3166-1, Codes for the Representation of Names of Countries and Their Subdivisions—Part 1: Country Codes.

ISO 8601, Date and time—Representations for information interchange.

SemVer, Semantic Versioning 1.0.0.[13]

[8]FIPS publications are available from the National Technical Information Service (https://www.ntis.gov/).
[9]The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc.
[10]IEEE publications are available from The Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (https://standards.ieee.org/).
[11]IETF documents (i.e. RFCs) are available for download at https://www.rfc-archive.org/.
[12]ISO publications are available from the ISO Central Secretariat (https://www.iso.org/). ISO publications are also available in the United States from the American National Standards Institute (https://www.ansi.org/).
[13]Available at: https://semver.org/spec/v1.0.0.html.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[14]

Activity: A type of Object making up the "this" in "I did this"; it is something with which an Actor interacted. It can be a unit of instruction, experience, or performance that is to be tracked in meaningful combination with a verb. Interpretation of Activity is broad, meaning that Activities can even be tangible objects such as a chair (real or virtual). In the Statement, "Anna tried a cake recipe," the recipe constitutes the Activity in terms of the Experience API (xAPI) Statement. Other examples of activities include a book, an e-learning course, a hike, or a meeting.

Activity Provider (AP): Now referred to as a Learning Record Provider (LRP). This change differentiates that the Activity itself is not always the responsibility of software, rather just that the tracking portion is.

Actor: An individual or group representation tracked using Statements performing an action within an Activity. The Actor is the "I" in "I did this."

Application Programming Interface (API): A set of rules and standards created to allow access into a software application or tool.

authentication: The concept of verifying identity. Authentication allows interactions between two "trusted" parties.

authorization: The affordance of permissions based on role; the process of making one party "trusted" by another.

client: Refers to any entity that might interact through requests. Some examples could be a Learning Record Provider, a Learning Record Consumer, a Learning Record Store (LRS), or a Learning Management System (LMS).

community of practice (CoP): 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 can also reuse existing vocabularies, profiles, and recipes already published by other CoPs or participants of the xAPI community.

document profile resource: A construct where information about the learner or activity is kept, typically in name/document pairs that have meaning to an instructional system component. Not to be confused with profile.

endpoint: An entry point in a service-oriented-architecture. xAPI mirrors this approach with document profile resources by defining the Internationalized Resource Identifier (IRI) from which communication takes place as an endpoint.

Experience API (xAPI): The collection of rules articulated in this document which determines how learning experiences are defined, formatted, and exchanged so that independent software programs can exchange and make use of this information.

[14] *IEEE Standards Dictionary Online* is available at: http://dictionary.ieee.org. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

immutable: Adjective used to describe things which cannot be changed. With some exceptions, Statements in the xAPI are immutable. This helps to ensure that when Statements are shared between Learning Record Stores (LRSs), multiple copies of the Statement remain the same.

Internationalized Resource Identifier (IRI): A unique identifier which could be an IRL. Used to identify an object such as a verb, activity, or activity type. Unlike Uniform Resource Identifiers (URIs), IRIs can contain some characters outside of the ASCII character set in order to support international languages. IRIs always include a scheme. This is not a requirement of this standard, but part of the definition of IRIs, per RFC 3987.[15] What are sometimes called "relative IRIs" are not IRIs.

Internationalized Resource Locator (IRL): In the context of this document, an IRL is an IRI that when translated into a Uniform Resource Identifier (URI) (per the IRI to URI rules), is a Uniform Resource Locator (URL).

Inverse Functional Identifier (IFI): An identifier which is unique to a particular persona or group.

learning experience: An event associated with learning. It is highly diverse as far as what it can be. Examples include reading a book, taking an online course, going on a field trip, engaging in self- directed research, or receiving a certificate for a completed course.

Learning Management System (LMS): A software package used to administer one or more courses to one or more learners. An LMS is typically a web-based system that allows learners to authenticate themselves, register for courses, complete courses and take assessments (Learning Systems Architecture Laboratory definition). An LMS in this document is used as an example of how a user is identified as "trusted" within a system and able to access its learning experiences.

Learning Record Consumer (LRC): An Experience API (xAPI) Client that accesses data from Learning Record Store(s) (LRS(s)) with the intent of processing the data, including interpretation, analysis, translation, dissemination, and aggregation.

Learning Record Provider (LRP): An Experience API (xAPI) Client that sends data to Learning Record Store(s) [LRS(s)]. Often, the LRP creates learning records while monitoring a learner as a part of a learning experience.

Learning Record Store (LRS): A server (i.e., a system capable of receiving and processing web requests) that is responsible for receiving, storing, and providing access to learning records.

learning record: An account of a learning experience that is formatted according to the rules of xAPI. A Learning Record takes on many forms, including Statements, documents, and their parts. This definition is intended to be all-inclusive.

metadata consumer: A person, organization, software program or other thing that seeks to determine the meaning represented by an Internationalized Resource Identifier (IRI) used within this specification and/or retrieves metadata about an IRI. A Learning Record Store (LRS) might or might not be a metadata consumer.

metadata provider: A person, organization, software program or other thing that coins Internationalized Resource Identifiers (IRIs) to be used within this specification and/or hosts metadata about an IRI.

persona: A set of one or more representations which defines an Actor uniquely. Conceptually, this is like having a "home email" and a "work email." Both are the same person, but have different data, associations, etc.

[15]Information on references can be found in Clause 2.

profile: A specific set of rules and documentation for implementing an Experience API (xAPI) in a particular context. A profile provides a way to talk about vocabulary concepts, statement templates, and patterns for xAPI data. Not to be confused with document profile resource.

registration: An instance of an Actor experiencing a particular Activity.

Representational State Transfer (REST): An architecture for designing networked web services. It relies on hypertext transfer protocol (HTTP) methods and uses current web best practices.

service: A software component responsible for one or more aspects of the distributed learning process.

Statement: A data structure showing evidence for any sort of experience or event which is to be tracked in Experience (xAPI) as a learning record. A set of several Statements, each representing an event in time, might be used to track complete details about a learning experience.

Verb: The action being done by the Actor within the Activity within a Statement. A Verb represents the "did" in "I did this."

3.2 Acronyms and abbreviations

AP	Activity Provider
API	Application Programming Interface
CoP	community of practice
IFI	Inverse Functional Identifier
IRI	Internationalized Resource Identifier
IRL	Internationalized Resource Locator
LRC	Learning Record Consumer
LRP	Learning Record Provider
LRS	Learning Record Store
REST	Representational State Transfer
URI	Uniform Resource Identifier
xAPI	Experience API

4. Learning Record Stores (LRSs)

Tracking in this standard is accomplished through HTTP Requests from the Learning Record Provider (LRP) to the Learning Record Store (LRS) (a server system with requirements defined in this document). Clause 4 details all requirements for the Learning Record Store. Accessing data in this standard is also accomplished through HTTP requests from a Learning Record Consumer (LRC).

4.1 LRS Communication

The primary function of the LRS within the xAPI is to store and retrieve Statements. Validation of Statements in the xAPI is focused solely on syntax, not semantics. Enforcing the rules that help ensure valid meaning among Verb definitions, Activity types, and extensions is the not the responsibility of the LRS, it should only enforce rules regarding structure.

The following table summarizes the Resources with which HTTP methods can interact. Implementation details for each resource can be found in 5.1.6. The LRS is interacted with via RESTful HTTP methods to the resources outlined in this clause.

Base Resource IRI/URL of the LRS Precedes Each Entry	Function	Supported Calls
statements	Statement Storage/Retrieval	PUT, POST, GET, HEAD
agents	Agent Object Storage/Retrieval	GET, HEAD
agents/profile	Agent Profile Resource	PUT, POST, GET, HEAD, DELETE
activities	Activity Object Storage/Retrieval	GET, HEAD
activities/profile	Activity Profile Resource	PUT, POST, GET, HEAD, DELETE
activities/state	State Resource	PUT, POST, GET, DELETE, HEAD
about	LRS Information	GET, HEAD
extensions/"yourext"	Any Additional Resource Not Identified in this Document	Not specified

The LRS *shall* enforce requirements as found in the tables in this clause, by rejecting any request with appropriate error code where invalid. Other Resources and undefined requests are out of scope of this document. It is suggested that an LRS *should not* reject those requests for the sole purpose of their absence from this document (but may certainly do so for security/unsupported reasons)

Examples of the resources and data requirements can be found at https://opensource.ieee.org/xapi/xapi-base-standard-examples.

4.1.1 Headers

The LRS *shall* implement and validate, per other requirements in this document, the following request headers:

- Accept
- Accept-Encoding
- Accept-Language
- Authorization
- Content-Type
- Content-Length
- Content-Transfer-Encoding
- If-Match
- If-None-Match
- X-Experience-API-Version

The following response headers are expected to be used by the LRS. Again, not all of these apply to every type of request and/or situation:

- Content-Type
- Content-Length

— Last-Modified

— ETag

— Status

— X-Experience-API-Version

— X-Experience-API-Consistent-Through

The lists above are not intended to be exhaustive of LRS requirements. Implementation details can be found throughout this document.

4.1.2 Encoding

All strings *shall* be encoded and interpreted as UTF-8 (RFC 3629).

4.1.3 Content types

Requests and responses within this specification normally use an application/json content type. Exceptions to this include the following:

— Documents can have any content type.

— Statement requests that can sometimes include Attachments use the multipart/mixed content type.

4.1.3.1 Application/JSON

— When receiving a PUT or POST with a document type of application/json, an LRS *shall* accept batches of Statements which contain no Attachment Objects.

— When receiving a PUT or POST with a document type of application/json, an LRS *shall* accept batches of Statements which contain only Attachment Objects with a populated fileUrl.

4.1.3.2 Multipart/Mixed

The multipart/mixed content type is used for requests that *could* include Attachments. This does not mean that all "multipart/mixed" requests necessarily do include Attachments.

4.1.3.3 Procedure for the exchange of attachments

— A Statement request including zero or more Attachments is construed in accordance with requirements in this document.

— The Statement is sent using a Content-Type of multipart/mixed. Any Attachments are placed at the end of such transmissions.

— The LRS decides whether to accept or reject the Statement based on the information in the first section. The Statement is sent using a Content-Type of multipart/mixed. Any Attachments are placed at the end of such transmissions.

— If it accepts the request, it can match the raw data of an Attachment(s) with the Attachment header by comparing the SHA-2 (FIPS PUB 180-2) of the raw data to the SHA-2 declared in the header. It *shall* not do so any other way.

4.1.3.4 Requirements for Attachment Statement Batches

A request transmitting a Statement batch, Statement results, or single Statement that includes Attachments *shall* satisfy one of the following criteria:

- It *shall* be of type application/json and include a fileUrl for every Attachment EXCEPT for Statement results when the "attachments" filter is false.
- It *shall* conform to the definition of "multipart/mixed" in RFC 2046 and:
  - The first part of the multipart document *shall* contain the Statements themselves, with type application/json.
  - Each additional part contains the raw data for an Attachment and forms a logical part of the Statement. This capability is available when issuing PUT or POST requests against the Statement Resource.
  - It *shall* include an X-Experience-API-Hash parameter in each part's header after the first (Statements) part.
  - It *shall* include a Content-Transfer-Encoding parameter with a value of binary in each part's header after the first (Statements) part.
  - It *should* only include one copy of an Attachment's data when the same Attachment is used in multiple Statements that are sent together.
  - It *should* include a Content-Type parameter in each part's header. For the first part (containing the Statement) this *shall* be application/json.
  - Where parameters have a corresponding property within the attachment Object (and both the parameter and property are specified for a given Attachment, the value of these parameters and properties *shall* match.

4.1.3.5 LRS Requirements

- An LRS *shall* include Attachments in the Transmission Format described above when issued a Request.
- When receiving a PUT or POST with a document type of multipart/mixed, an LRS *shall* accept batches of Statements that contain Attachments in the Transmission Format described above.
- When receiving a PUT or POST with a document type of multipart/mixed, an LRS *shall* reject batches of Statements having Attachments that neither contain a fileUrl nor match a received Attachment part based on their hash.
- When receiving a PUT or POST with a document type of multipart/mixed, an LRS *should* assume a Content-Transfer-Encoding of binary for Attachment parts.
- When receiving a PUT or POST with a document type of multipart/mixed, an LRS *shall* accept batches of Statements which contain no Attachment Objects.
- When receiving a PUT or POST with a document type of multipart/mixed, an LRS *shall* accept batches of Statements which contain only Attachment Objects with a populated fileUrl.

NOTE—There is no requirement that Statement batches using the "mime/multipart" format contain Attachments.[16]

[16]Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

The File uniform resource locator (URL) is intended to provide a location from which the Attachment can be received. There are, however, no requirements for the owner of the Attachment to make the Attachment data available at the location indefinitely or to make the Attachment publicly available without security restrictions. When determining Attachment hosting arrangements, those creating Statements using the "fileUrl" property are encouraged to consider the needs of end recipient(s) of the Statement, especially if the Attachment content is not included with the Statement.

The period of time an Attachment is made available for, and the security restrictions applied to hosted attachments, are out of scope of this specification.

4.1.4 Concurrency

Concurrency control makes certain that a PUT, POST, or DELETE does not perform operations based on old data.

xAPI uses HTTP 1.1 entity tags (ETags) to implement optimistic concurrency control in the following resources, where PUT, POST or DELETE are allowed to overwrite or remove existing data:

- State Resource
- Agent Profile Resource
- Activity Profile Resource

LRS Requirements

- An LRS responding to a GET request *shall* add an ETag HTTP header to the response.
- An LRS responding to a PUT, POST, or DELETE request *shall* handle the "If-Match" header as described in RFC2616, HTTP 1.1 in order to detect modifications made after the document was last fetched.

If the header precondition in either of the request cases above fails, the LRS:

- *shall* return HTTP status 412 Precondition Failed
- *shall not* make a modification to the resource

If a PUT request is received without either header for a resource that already exists, the LRS:

- *shall* return HTTP status 409 Conflict
- *shall* return a response explaining that the Learning Record Provider *should*
  - check the current state of the resource
  - set the "If-Match" header with the current ETag to resolve the conflict
- *shall not* make a modification to the resource

4.1.5 Error Codes

The list below covers many error conditions that could be returned from various methods in the Application Programming Interface (API).

- 400 Bad Request: Indicates an error condition caused by an invalid or missing argument. The term "invalid arguments" includes malformed JSON (RFC 8259) or invalid Object structures.
- 401 Unauthorized: Indicates that authentication is required, or in the case authentication has been posted in the request, that the given credentials have been refused.
- 403 Forbidden: Indicates that the request is unauthorized for the given credentials. Note this is different than refusing the credentials given. In this case, the credentials have been validated, but the authenticated system is not allowed to perform the given action.
- 404 Not Found: Indicates the requested resource was not found. May be returned by any method that returns a uniquely identified resource, for instance, any State, Agent Profile, or Activity Profile Resource request targeting a specific document, or the method to retrieve a single Statement.
- 409 Conflict: Indicates an error condition due to a conflict with the current state of a resource, in the case of State Resource, Agent Profile Resource or Activity Profile Resource requests, or in the Statement Resource PUT or POST calls.
- 412 Precondition Failed: Indicates an error condition due to a failure of a precondition posted with the request, in the case of State or Agent Profile or Activity Profile API requests.
- 413 Request Entity Too Large: Indicates that the LRS has rejected the Statement or document because its size (or the size of an Attachment included in the request) is larger than the maximum allowed by the LRS.
- 429 Too Many Requests: Indicates that the LRS has rejected the request because it has received too many requests from the system or set of credentials in a given amount of time.
- 500 Internal Server Error: Indicates a general error condition, typically an unexpected exception in processing on the server.

Requirements

- An LRS *shall* return the error code most appropriate to the error condition from the list above.
- An LRS *should* return a message in the response explaining the cause of the error.
- An LRS *should* use content negotiation as described in RFC 7231 to decide the format of the error.
- An LRS *should* allow for plain text, HTML, and JSON (RFC 8259) responses for errors (using content negotiation).
- The LRS *shall* reject with 400 Bad Request status any requests that use any parameters which the LRS does not recognize in their intended context in this specification.

  NOTE—LRSs *may* recognize and act on parameters not in this specification.
- The LRS *shall* reject with 400 Bad Request status any requests that use any parameters matching parameters described in this specification in all but case.
- The LRS *shall* reject a batch of statements if any Statement within that batch is rejected.
- The LRS *shall* reject with 403 Forbidden status any request rejected by the LRS where the credentials associated with the request do not have permission to make that request.
- The LRS *shall* reject with 413 Request Entity Too Large status any request rejected by the LRS where the size of the Attachment, Statement or document is larger than the maximum allowed by the LRS.

— The LRS *may* choose any Attachment, Statement and document size limits and *may* vary this limit on any basis, e.g., per authority.

— The LRS *shall* reject with 429 Too Many Requests status any request rejected by the LRS where the request is rejected due to too many requests being received by a particular system or set of credentials in a given amount of time.

— The LRS *may* choose any rate limit and *may* vary this limit on any basis, e.g., per authority.

4.1.6 Resources

Each LRS Resource is described in the following subclauses. Each Resource has requirements for each of the HTTP methods that may be made to it. Each subclause includes the expected contents of the body of the request, the request parameters, and the expected returns. If an HTTP method is not described, it is out of scope of this document.

LRSs leverage both Statements and documents as data structures. The LRS *shall* support all of the resources described in this subclause.

— The LRS *may* support additional resources not described in this specification.

— Past, current and future versions of this specification do not define resource locations (endpoints) with path segments starting with extensions/. LRSs supporting additional resources not defined in this specification *should* define their resource locations with path segments starting with extensions/.

— A Learning Record Provider *may* send documents to any of the Document Resources for Activities and Agents that the LRS does not have prior knowledge of.

— The LRS *shall not* reject documents on the basis of not having prior knowledge of the Activity and/or Agent.

The LRS shall reject Requests where parameters are:

— Missing any required properties

— With any null values (except inside extensions)

— Where the wrong data type is used (see tables), for example:

— With strings where numbers are required, even if those strings contain numbers, or:

  — With strings where booleans are required, even if those strings contain booleans

  — With any non-format-following key or value, including the empty string, where a string with a particular format (such as mailto IRI, UUID, or IRI) is required

— Where the case of a key does not match the case specified in this specification

— Where the case of a value restricted to enumerated values does not match an enumerated value given in this specification exactly

— Where a key or value is not allowed by this specification

— Containing Internationalized Resource Locator (IRL) or Internationalized Resource Identifier (IRI) values without a scheme

NOTE—In all of the examples in this specification, http://example.com/xAPI/ is the example base resource location (endpoint) of the LRS. All other IRI syntax after this represents the particular resource used.

Note that the following table shows generic properties, not a JSON Object as many other tables in this specification do. The id is stored in the IRL, "updated" is HTTP header information, and "contents" is the HTTP document itself (as opposed to an Object).

Property	Type	Description
id	String	Set by Learning Record Provider, unique within the scope of the Agent or Activity.
updated	Timestamp	When the document was most recently modified (HTTP Header).
contents	Arbitrary binary data	The contents of the document

Examples of the resources can be found at https://opensource.ieee.org/xapi/xapi-base-standard- examples.

4.1.6.1 Statement Resource (/statements)

Statements are the key data structure of xAPI. This resource facilitates their storage and retrieval.

Example resource location/endpoint: http://example.com/xAPI/statements

4.1.6.1.1 PUT Request

Summary: Stores a single Statement with the given id.

Parameter	Type	Default	Description	Required
statementId	String		Id of Statement to record	Required

Body: The Statement object to be stored.

Returns: 204 No Content

- The LRS *may* respond before Statements that have been stored are available for retrieval.
- An LRS *shall not* make any modifications to its state based on receiving a Statement with a statementId that it already has a Statement for. Whether it responds with 409 Conflict or 204 No Content, it *shall not* modify the Statement or any other Object.
- If the LRS receives a Statement with an id it already has a Statement for, it *should* verify the received Statement matches the existing one and *should* return 409 Conflict if they do not match.

4.1.6.1.2 POST Request

Summary: Stores a Statement, or a set of Statements.

Body: An array of Statements or a single Statement to be stored.

Returns: 200 OK, Array of Statement id(s) (UUID) in the same order as the corresponding stored Statements.

- The LRS *may* respond before Statements that have been stored are available for retrieval.
- An LRS *shall not* make any modifications to its state based on receiving a Statement with an id that it already has a Statement for. Whether it responds with 409 Conflict or 204 No Content, it *shall not* modify the Statement or any other Object.
- If the LRS receives a Statement with an id it already has a Statement for, it *should* verify the received Statement matches the existing one and *should* return 409 Conflict if they do not match.

— If the LRS receives a batch of Statements containing two or more Statements with the same id, it *shall* reject the batch and return 400 Bad Request.

4.1.6.1.3 GET Request

Summary: Fetches a single Statement or multiple Statements. If the statementId or voidedStatementId parameter is specified a single Statement is returned. Otherwise, a StatementResult Object, a list of Statements in reverse chronological order based on "stored" time, subject to permissions and maximum list length. If additional results are available, an IRL to retrieve them is included in the StatementResult Object.

Parameter	Type	Default	Description	Required
statementId	String		Id of Statement to fetch.	Optional
voidedStatementId	String		Id of voided Statement to fetch.	Optional
agent	Agent or Identified Group Object (JSON)		Filter, only return Statements for which the specified Agent or Group is the Actor or Object of the Statement. Agents or Identified Groups are equal when the same Inverse Functional Identifier is used in each Object compared and those Inverse Functional Identifiers have equal values. For the purposes of this filter, Groups that have members which match the specified Agent based on their Inverse Functional. Identifier as described above are considered a match	Optional
verb	Verb id (IRI)		Filter, only return Statements matching the specified Verb id.	Optional
activity	Activity id (IRI)		Filter, only return Statements for which the Object of the Statement is an Activity with the specified id.	Optional
registration	UUID		Filter, only return Statements matching the specified registration id. Note that although frequently a unique registration is used for one Actor assigned to one Activity, this cannot be assumed. If only Statements for a certain Actor or Activity are required, those parameters also need to be specified	Optional
related_activities	Boolean	false	Apply the Agent filter broadly. Include Statements for which the Actor, Object, Authority, Instructor, Team, Context Agent, Context Group, or any of these properties in a contained SubStatement match the Agent parameter, instead of that parameter's normal behavior. Matching is defined in the same way it is for the "agent" parameter.	Optional

Table continues

Parameter	Type	Default	Description	Required
related_agents	Boolean	false	Apply the Agent filter broadly. Include Statements for which the Actor, Object, Authority, Instructor, Team, Context Agent, Context Group, or any of these properties in a contained SubStatement match the Agent parameter, instead of that parameter's normal behavior. Matching is defined in the same way it is for the "agent" parameter.	Optional
since	Timestamp		Only Statements stored since the specified Timestamp (exclusive) are returned.	Optional
until	Timestamp		Only Statements stored at or before the specified Timestamp are returned.	Optional
limit	Nonnegative Integer	0	Maximum number of Statements to return. 0 indicates return the maximum the server allows.	Optional
format	String: (ids, exact, or canonical)	exact	If ids, only include minimum information necessary in Agent, Activity, Verb and Group Objects to identify them. For Anonymous Groups this means including the minimum information needed to identify each member. If exact, return Agent, Activity, Verb and Group Objects populated exactly as they were when the Statement was received. An LRS requesting Statements for the purpose of importing them would use a format of "exact" in order to maintain Statement Immutability. If canonical, return Activity Objects and Verbs populated with the canonical definition of the Activity Objects and Display of the Verbs as determined by the LRS, after applying the "Language Filtering Requirements for Canonical Format Statements", and return the original Agent and Group Objects as in "exact" mode.	Optional
attachments	Boolean	false	If true, the LRS uses the multipart response format and includes all attachments as described previously. If false, the LRS sends the prescribed response with Content-Type application/json and does not send attachment data.	Optional
ascending	Boolean	false	If true, return results in ascending order of stored time.	Optional

Body: None.

Returns: 200 OK, Statement or Statement Result

NOTE—The values of Boolean parameters are represented as true or false as in JSON.

— The LRS *shall* reject with a 400 Bad Request error any requests to this resource which contain both statementId and voidedStatementId parameters.

— The LRS *shall* reject with a 400 Bad Request error any requests to this resource which contain statementId or voidedStatementId parameters, and also contain any other parameter besides "attachments" or "format".

— The LRS *may* apply additional query filter criteria based on permissions associated with the credentials used.

— In the event that no Statements are found matching the query filter criteria, the LRS *shall* stil. return 200 OK and a StatementResult Object. In this case, the "statements" property *shall* contain an empty array.

— The LRS *shall* include the header "X-Experience-API-Consistent-Through", in ISO 8601 combined date and time format, on all responses to Statements Resource requests, with a value of the timestamp for which all Statements that have a "stored" property before that time are known with reasonable certainty to be available for retrieval. This time *should* take into account any temporary condition, such as excessive load, which might cause a delay in Statements becoming available for retrieval. It is expected that this is a recent timestamp, even if there are no recently received Statements.

— If the "attachment" property of a GET Statement is used and is set to true, the LRS *shall* use the multipart response format and include all Attachments as described in this document.

— If the "attachment" property of a GET statement is used and is set to false, the LRS *shall not* include Attachment raw data and *shall* report application/json.

— The LRS *shall* include, in the "Last-Modified" header, the most recent (maximum) "stored" property of any of the returned statement(s).

— When queried for Statements with a Format of exact, the LRS *shall* return the "display" property exactly as included (or omitted) within the Statement.

4.1.6.1.4 Filter Conditions for StatementRefs

This subclause outlines rules by which Statements targeting other Statements can sometimes be considered to meet the filter conditions of a query even if they do not match the original query's filter parameters. These rules *do not* apply when retrieving a single Statement using "statementId" or "voidedStatementId" query parameters.

'Targeting Statements' means that one Statement (the targeting Statement) includes the Statement id of another Statement (the targeted Statement) as a Statement Reference as the Object of the Statement.

For filter parameters which are not time or sequence based (that is, other than "since", "until", or "limit"), Statements which target another Statement (by using a StatementRef as the Object of the Statement) meet the filter condition if the targeted Statement meets the filter condition.

The time and sequence-based parameters *shall* still be applied to the Statement making the StatementRef in this manner. This rule applies recursively, so that "Statement a" is a match when a targets b which targets c and the filter conditions described above match for "Statement c".

For example, consider the Statement 'Ben passed explosives training', and a follow up Statement: "Andrew confirmed <StatementRef to original Statement>". The follow up Statement does not mention "Ben" or "explosives training," but when fetching Statements with an Actor filter of "Ben" or an Activity filter of "explosives training," both Statements match and are returned so long as they fall into the time or sequence being fetched.

NOTE—StatementRefs used as a value of the "Statement" property within Context do not affect how Statements are filtered.

4.1.6.1.5 Language Filtering Requirements for Canonical Format Statements

— Activity Objects contain Language Map Objects within their "name", "description" and various interaction components. The LRS *shall* return only one language in each of these maps.

— The LRS *may* maintain canonical versions of language maps against any IRI identifying an object containing language maps. This includes the language map stored in the Verb's "display" property and potentially some language maps used within extensions.

— The LRS *may* maintain a canonical version of any language map and return this when canonical format is used to retrieve Statements. The LRS *shall* return only one language within each language map for which it returns a canonical map.

— In order to choose the most relevant language, the LRS *shall* apply the "Accept-Language" header as described in RFC 2616 (HTTP 1.1), except that this logic *shall* be applied to each language map individually to select which language entry to include, rather than to the resource (list of Statements) as a whole.

4.1.6.1.6 Voided Statements

— The LRS *shall not* return any Statement which has been voided, unless that Statement has been requested by voidedStatementId. The previously described process is no exception to this requirement. The process of retrieving voiding Statements is to request each individually by voidedStatementId.

— The LRS *shall* still return any Statements targeting the voided Statement, following the process and conditions previously described. This includes the voiding Statement, which cannot be voided.

4.1.6.2 State Resource (/activities/state)

A place to store information about the state of an activity in a generic form called a document. The intent of this resource is to store/retrieve a specific agent's data within a specific activity, potentially tied to a registration. The semantics of the LRS response are driven by the presence of a "stateId" parameter. If it is included, the GET and DELETE methods act upon a single defined state document identified by "stateId". Otherwise, GET returns the available ids, and DELETE deletes all state in the context given through the other parameters. This resource has concurrency controls associated with it.

Example resource location/endpoint: http://example.com/xAPI/activities/state

4.1.6.2.1 PUT Request

Summary: Stores a single document with the given id.

Body: The document object to be stored.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent Object (JSON)	The Agent associated with this state.	Required
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required

4.1.6.2.2 POST Request

Summary: Updates/stores a single document with the given id.

Body: The document object to be stored/updated.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent Object (JSON)	The Agent associated with this state.	Required
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required

When an LRS receives a POST request with content type application/json for an existing document also of content type application/json, it *shall* merge the posted document with the existing document. In this context, merge is defined as:

— De-serialize the Objects represented by each document

— For each property directly defined on the Object being posted, set the corresponding property on the existing Object equal to the value from the posted Object

— Store any valid json serialization of the existing Object as the document referenced in the request

Note that only top-level properties are merged, even if a top-level property is an Object. The entire contents of each original property are replaced with the entire contents of each new property.

— If the document being posted or any existing document does not have a Content-Type of application/json, or if either document cannot be parsed as a JSON Object, the LRS *shall* respond with HTTP status code 400 Bad Request, and *shall not* update the target document as a result of the request.

— If the merge is successful, the LRS *shall* respond with HTTP status code 204 No Content.

4.1.6.2.3 (Single Document) GET Request

Summary: Fetches a single document with the given id.

Body: None.

Returns: 200, The State Document

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent	The Agent	Required
	Object (JSON)	associated with this state.	
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required unless since parameter is present. In that case, Not Allowed.
since	Timestamp	Do not use for single document GET request.	Optional if stateId is not used. If the stateId parameter is included, Not Allowed.

The LRS *shall* include a "Last-Modified" header indicating when the document was last modified.

4.1.6.2.4 (Single Document) DELETE Request

Summary: Deletes a single document with the given id.

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent Object (JSON)	The Agent associated with this state.	Required
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required

4.1.6.2.5 (Multiple Document) GET Request

Summary: Fetches State ids of all state data for this context (Activity + Agent [+ registration if specified]). If "since" parameter is specified, this is limited to entries that have been stored or updated since the specified timestamp (exclusive).

Body: None.

Returns: 200, Array of State ids

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with these states.	Required
agent	Agent object (JSON)	The Agent associated with these states.	Required
registration	UUID	The Registration associated with these states.	Optional
stateId	String	Do not use for Multiple Document GET request.	Required unless since parameter is present. In that case, Not Allowed.
since	Timestamp	Only ids of states stored since the specified Timestamp (exclusive) are returned.	Optional as long as stateId is not used. If the stateId parameter is included, Not Allowed.

4.1.6.2.6 (Multiple Document) DELETE Request

Summary: Deletes all state data for this context (Activity + Agent [+ registration if specified]).

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with these states.	Required
agent	Agent object (JSON)	The Agent associated with these states.	Required
registration	UUID	The Registration associated with these states.	Optional

4.1.6.3 Agents Resource (/agents)

The Agents Resource provides a method to retrieve a special Object with combined information about an Agent derived from an outside service, such as a directory service. This object is called a "Person Object". The Person Object is very similar to an Agent Object, but instead of each attribute having a single value, each attribute has an array value, and it is legal to include multiple identifying properties. This is different from the FOAF concept of person, person is being used here to indicate a person- centric view of the LRS Agent data, but Agents just refer to one persona (a person in one context).

Example resource location/endpoint: http://example.com/xAPI/agents

4.1.6.3.1 GET Request

Summary: Return a Person Object for a specified Agent.

Body: None.

Returns: 200 OK, Person Object

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent representation to use in fetching expanded Agent information.	Required

4.1.6.3.2 Person Object Details

Property	Type	Description	Required
objectType	String	Person	Required
name	Array of strings.	List of names of Agents retrieved.	Optional
mbox	Array of IRIs in the form "mailto:email address".	List of e-mail addresses of Agents retrieved.	Optional
mbox_sha1sum	Array of strings.	List of the SHA1 hashes of mailto IRIs (such as go in an mbox property).	Optional
openid	Array of strings.	List of openids that uniquely identify the Agents retrieved.	Optional
account	Array of account objects.	List of accounts to match. Complete account Objects (homePage and name) *shall* be provided.	Optional

— If an LRS does not have any additional information about an Agent to return, the LRS *shall* still return a Person Object when queried, but that Person Object only includes the information associated with the requested Agent.

— All array properties *shall* be populated with members with the same definition as the similarly named property from Agent Objects.

4.1.6.4 Activities Resource (/activities)

The Activities Resource provides a method to retrieve a full description of an Activity from the LRS.

Example resource location/endpoint: http://example.com/xAPI/activities

4.1.6.4.1 GET Request

Summary: Fetches the complete Activity Object specified.

Body: None.

Returns: 200 OK, Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The id associated with the Activities to load.	Required

If an LRS does not have a canonical definition of the Activity to return, the LRS *shall* still return an Activity Object when queried.

4.1.6.5 Agent Profile Resource (/agents/profile)

A place to store information about an Agent in a generic form called a document. This information is not tied to an activity or registration. The semantics of the LRS response are driven by the presence of a "profileId" parameter. If it is included, the GET method acts upon a single defined profile document identified by "profileId". Otherwise, GET returns the available ids given through the other parameter. This resource has concurrency controls associated with it.

Example resource location/endpoint: http://example.com/xAPI/agents/profile

4.1.6.5.1 PUT Request

Summary: Stores a single document with the given id.

Body: The document to be stored.

Returns: 204 No Content

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

4.1.6.5.2 POST Request

Summary: Stores/updates a single document with the given id.

Body: The document to be stored/updated.

Returns: 204 No Content

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

When an LRS receives a POST request with content type application/json for an existing document also of content type application/json, it *shall* merge the posted document with the existing document. In this context, merge is defined as:

— de-serialize the Objects represented by each document.

— for each property directly defined on the Object being posted, set the corresponding property on the existing Object equal to the value from the posted Object.

— store any valid json serialization of the existing Object as the document referenced in the request.

Note that only top-level properties are merged, even if a top-level property is an Object. The entire contents of each original property are replaced with the entire contents of each new property.

— If the document being posted or any existing document does not have a Content-Type of application/json, or if either document cannot be parsed as a JSON Object, the LRS *shall* respond with HTTP status code 400 Bad Request, and *shall not* update the target document as a result of the request.

— If the merge is successful, the LRS *shall* respond with HTTP status code 204 No Content.

4.1.6.5.3 DELETE Request

Summary: Deletes a single document with the given id.

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

4.1.6.5.4 (Single Document) GET Request

Summary: Fetches a single document with the given id.

Body: None.

Returns: 200 OK, the Profile document

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

The LRS *shall* include a “Last-Modified” header indicating when the document was last modified.

4.1.6.5.5 (Multiple Document) GET Request

Summary: Fetches Profile ids of all Profile documents for an Agent. If "since" parameter is specified, this is limited to entries that have been stored or updated since the specified Timestamp (exclusive).

Body: None.

Returns: 200 OK, Array of profileIds.

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent associated with this Profile document.	Required
since	Timestamp	Only ids of Profiles stored since the specified Timestamp (exclusive) are returned.	Optional

4.1.6.6 Activity Profile Resource (/activities/profile)

A place to store information about an Activity in a generic form called a document. This information is not tied to an Actor or registration. The semantics of the LRS response are driven by the presence of a "profileId" parameter. If it is included, the GET method acts upon a single defined profile document identified by "ProfileId". Otherwise, GET returns the available ids given through the other parameter.

This resource has concurrency controls associated with it.

Example resource location/endpoint: http://example.com/xAPI/activities/profile

4.1.6.6.1 PUT Request

Summary: Stores a single document with the given id.

Body: The document to be stored.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

4.1.6.6.2 POST Request

Summary: Stores/updates a single document with the given id.

Body: The document to be stored/updated.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

When an LRS receives a POST request with content type application/json for an existing document also of content type application/json, it *shall* merge the posted document with the existing document. In this context, merge is defined as:

— De-serialize the Objects represented by each document

— For each property directly defined on the Object being posted, set the corresponding property on the existing Object equal to the value from the posted Object

— Store any valid json serialization of the existing Object as the document referenced in the request

Note that only top-level properties are merged, even if a top-level property is an Object. The entire contents of each original property are replaced with the entire contents of each new property.

— If the document being posted or any existing document does not have a Content-Type of application/json, or if either document cannot be parsed as a JSON Object, the LRS *shall* respond with HTTP status code 400 Bad Request, and *shall not* update the target document as a result of the request.

— If the merge is successful, the LRS *shall* respond with HTTP status code 204 No Content.

4.1.6.6.3 DELETE Request

Summary: Deletes a single document with the given id.

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

4.1.6.6.4 (Single Document) GET Request

Summary: Fetches a single document with the given id.

Body: None.

Returns: 200 OK, the Profile document

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

The LRS *shall* include a “Last-Modified” header indicating when the document was last modified.

4.1.6.6.5 (Multiple Document) GET Request

Summary: Fetches Profile ids of all Profile documents for an Activity. If "since" parameter is specified, this is limited to entries that have been stored or updated since the specified Timestamp (exclusive).

Body: None.

Returns: 200 OK, Array of profileIds.

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent associated with this Profile document.	Required
since	Timestamp	Only ids of Profiles stored since the specified Timestamp (exclusive) are returned.	Optional

4.1.6.7 About Resource (/about)

Returns JSON Object containing information about this LRS, including supported extensions and xAPI version(s).

Example resource location/endpoint: http://example.com/xAPI/about

4.1.6.7.1 GET Request

Body: None.

Returns: 200 OK, JSON object containing basic metadata about this LRS

Property	Type	Description	Required
version	Array of version strings	xAPI versions this LRS supports	Required
extensions	Object	A map of other properties as needed or supported additional resources (extensions)	Optional

- An LRS *shall* return the JSON document described above, with a "version" property that includes the latest minor and patch version the LRS conforms to, for each major version.
- An LRS *shall not* reject requests based to this resource on their version header as would otherwise be required by Versioning.

4.1.7 Versioning

Using Semantic Versioning allows LRPs and LRSs to remain interoperable as the specification changes.

4.1.7.1 Details

xAPI is versioned according to Semantic Versioning 1.0.0. Every request from an LRP (or other systems that may interact with an LRS) and every response from the LRS includes an HTTP header with the name X-Experience-API-Version and the version as the value. For example, X-Experience-API-Version: 2.0.0 for version 2.0.0;

NOTE—For patch versions of the specification later than 2.0.0, the "X-Experience-API-Version" header does not match the statement version property which is always 2.0.0 for all 2.0.x versions of the spec. The

"X-Experience-API-Version" header enables the LRS and client system to determine the exact patch version of the specification being followed. While no communication incompatibility should arise among 2.0.x versions, there are sometimes clarifications of previously intended behavior.

4.1.7.2 LRS Requirements

- The LRS *shall* include the "X-Experience-API-Version" header in every response.
- The LRS *shall* set this header to the latest patch version.
- The LRS *shall* accept requests with a version header of 2.0 as if the version header was 2.0.0.
- The LRS *shall* reject requests with version header prior to version 2.0.0 unless such requests are routed to a fully conformant implementation of a prior version specified in the header.
- The LRS *shall* reject requests with a version header of 2.1.0 or greater.
- The LRS *shall* make these rejects by responding with a 400 Bad Request error including a short description of the problem.

4.1.8 Authentication

4.1.8.1 Rationale

In order to support the varying security requirements of different environments, a specific authentication mechanism is not defined.

4.1.8.2 Requirements

The LRS *shall* handle making, or delegating, decisions on the validity of Statements, and determining what operations might be performed based on the credentials used (or lack thereof).

4.1.9 Security

Security is beyond the scope of this document and left to the individual LRS provider as an implementation detail. Implementors are encouraged to follow industry best practices, e.g., The HTTPS-Only Standard from the office of the White House CIO.

It is possible that security concerns may arise in the implementation of this specification, and implementers might choose to break a conformance requirement for the sake of security. In these cases, implementers are encouraged to consider both the security and interoperability implications of their implementation decisions. In any case, the LRS still needs to be configurable such that it is able to pass conformance tests.

4.2 LRS Data Requirements

The LRS is responsible for handling various data formats described in 1.1. This includes following acceptance criteria for those data formats. The most common data format in xAPI is the Statement.

The required field in each of these tables dictates the LRS responsibility in validating data. Required indicates the LRS needs this property and rejects Statements that do not have them. Recommended indicates that the LRS *should* be given this property. Optional indicates that the field may be used as is and the LRS takes no default responsibility if it is not provided. Not Recommended indicates that the LRS *should not* receive this property and instead the LRS should populate the value.

Statement Immutability:

While the methods to PUT/POST and GET Statements are specific, the storage mechanism is not. JSON has flexibility in form, which means that a reconstruction may not be exact, ordering being a very common example. This document defines only certain aspects of Statements to be immutable. For all intents and purposes, if immutability required fields are equal, then the Statements are equal.

- The LRS *shall* store, retrieve, and compare Statements in a way that preserves immutability requirements, which includes the following properties with exceptions in parentheses"
  - Actor (except the ordering of group members)
  - Verb (except for display)
  - Object
  - Duration (see 4.2.7.6 for further requirements)
- The LRS *shall* specifically *not* consider any of the following for equivalence, nor is it responsible for preservation as described above for the following properties/cases:
  - Case (upper versus lower)
  - Id
  - Order of any Group Members ○ Authority
  - Stored
  - Timestamp
  - Version
  - Any attachments
  - Any referenced Activity Definitions

Examples of the data requirements can be found at https://opensource.ieee.org/xapi/xapi-base-standard-examples.

4.2.1 Table Guidelines

Tables in this document represent requirements that *shall* be followed. It is the responsibility of an LRS to reject requests that contain data that does not follow requirements in these tables. The following requirements are expected to be followed for each table in this subclause.

NOTE—The "description" portion of table has requirements, in particular for controlled vocabulary, enumerations, etc.

a) The LRS *shall* reject any Statement that does not conform to the "Type" field.

b) An LRS *shall* reject a Statement with additional properties other than extensions in the locations where extensions are allowed

c) An LRS shall reject a Statement with invalid IRIs

d) An LRS shall reject a Statement that uses a property more than once

e) The LRS shall reject Statements:

   1) Missing any required properties

   2) With any null values (except inside extensions).

3) Where the wrong data type is used (see tables), for example:
   i) With strings where numbers are required, even if those strings contain numbers, or
   ii) With strings where booleans are required, even if those strings contain booleans
4) With any non-format-following key or value, including the empty string, where a string with a particular format (such as mailto IRI, UUID, or IRI) is required
5) Where the case of a key does not match the case specified in this specification
6) Where the case of a value restricted to enumerated values does not match an enumerated value given in this specification exactly
7) Where a key or value is not allowed by this specification
8) Containing IRL or IRI values without a scheme

4.2.2 Statement

Each Statement in xAPI has the following JSON structure and requirements. Note that requirements for properties with type "Object" are included in subsequent tables.

Property	Type	Description	Required
id	UUID	UUID assigned by LRS if not set by the Learning Record Provider.	Recommended
actor	Object	Whom the Statement is about, as an Agent or Group Object.	Required
verb	Object	Action taken by the Actor.	Required
object	Object	Activity, Agent, or another Statement that is the Object of the Statement.	Required
result	Object	Result Object, further details representing a measured outcome.	Optional
context	Object	Context that gives the Statement more meaning. Examples: a team the Actor is working with, altitude at which a scenario was attempted in a flight simulator.	Optional
timestamp	Timestamp	Timestamp of when the events described within this Statement occurred (it can represent any point during an experience, not necessarily the beginning or end). Set by the LRS if not provided.	Optional
stored	Timestamp	Timestamp of when this Statement was recorded. Set by LRS.	Not Recommended
authority	Object	Agent or Group who is asserting this Statement is true. Verified by the LRS based on authentication. Set by LRS if not provided or if a strong trust relationship between the Learning Record Provider and LRS has not been established.	Optional
version	Version	The Statement's associated xAPI version, formatted according to Semantic Versioning 1.0.0.	Not Recommended
attachments	Ordered array of Attachment Objects	Headers for Attachments to the Statement	Optional

4.2.2.1 Actor

Actors can take on the form of either Groups or Agents. Groups can be identified, and if they are not, are considered to be anonymous groups. Specific requirements are found in the tables below.

Actor as Agent Table: The table below lists the properties of Agent Objects.

Property	Type	Description	Required
objectType	String	Value is "Agent". This property is optional except when the Agent is used as a Statement's object.	Optional (except when it is to be used in a Statement Object)
name	String	Full name of the Agent.	Optional
mbox	mailto IRI	The required format is "mailto:email address". Only email addresses uniquely assigned to this Agent *should* be used for this property and mbox_sha1sum.	Exactly One of mbox, mbox_sha1sum, openid, account is required
mbox_sha1sum	String	The hex-encoded SHA1 hash of a mailto IRI (i.e., the value of an mbox property). An LRS *may* include Agents with a matching hash when a request is based on an mbox.	Exactly One of mbox, mbox_sha1sum, openid, account is required
openid	URI	An openID that uniquely identifies the Agent.	Exactly One of mbox, mbox_sha1sum, openid, account is required
account	Object	A user account on an existing system e.g., an LMS or intranet.	Exactly One of mbox, mbox_sha1sum, openid, account is required

Actor as Anonymous Group Table: The table below lists all properties of an Anonymous Group.

Property	Type	Description	Required
objectType	String	Group.	Required
name	String	Name of the Group.	Optional
member	Array of Agent Objects	The members of this Group. This is an unordered list.	Required

Actor as Identified Group Table: The table below lists all properties of an Identified Group.

Property	Type	Description	Required
objectType	String	Group.	Required
name	String	Name of the Group.	Optional
member	Array of Agent Objects	The members of this Group. This is an unordered list.	Optional
mbox	mailto IRI	The required format is "mailto:email address". Only email addresses that have only ever been assigned to this Agent, but no others, *should* be used for this property and mbox_sha1sum.	Exactly One of mbox, mbox_sha1sum, openid, account is required
mbox_sha1sum	String	The hex-encoded SHA1 hash of a mailto IRI (i.e., the value of an mbox property). An LRS *may* include Agents with a matching hash when a request is based on an mbox.	Exactly One of mbox, mbox_sha1sum, openid, account is required
openid	URI	An openID that uniquely identifies the Agent.	Exactly One of mbox, mbox_sha1sum, openid, account is required
account	Object	A user account on an existing system e.g., an LMS or intranet.	Exactly One of mbox, mbox_sha1sum, openid, account is required

Account Table: The table below lists all properties of Account Objects.

Property	Type	Description	Required
homePage	IRL	The canonical home page for the system the account is on. This is based on FOAF's accountServiceHomePage.	Required
name	String	The unique id or name used to log in to this account. This is based on FOAF's accountName.	Required

4.2.2.2 Verb

Verbs appear in Statements as Objects consisting of an IRI and a set of display names corresponding to multiple languages or dialects which provide human-readable meanings of the Verb. The table below lists all properties of the Verb Object.

Property	Type	Description	Required
id	IRI	Corresponds to a Verb definition. Each Verb definition corresponds to the meaning of a Verb, not the word.	Required
display	Language Map	The human readable representation of the Verb in one or more languages. This does not have any impact on the meaning of the Statement, but serves to give a human-readable display of the meaning already determined by the chosen Verb.	Recommended

4.2.2.3 Object

Objects in xAPI vary widely in use and are classified in that use by the objectType property. Each of the following subclauses provides additional requirements that stem from the use of objectType.

The LRS shall treat any Object without an objectType as though the value was "Activity".

Object As Activity Table: The following table lists the Object properties of an Activity.

Property	Type	Description	Required
objectType	String	*shall* be Activity when present	Optional
id	IRI	An identifier for a single unique Activity	Required
definition	Object	Metadata	Optional

Activity Definition (referenced in Object as Activity) Table: All properties are optional, but if implemented, some properties in sub-properties are required.

Property	Type	Description	Required
name	Language Map	The human readable/visual name of the Activity	Recommended
description	Language Map	A description of the Activity	Recommended
type	IRI	The type of Activity.	Recommended
moreInfo	IRL	Resolves to a document with human-readable information about the Activity, which could include a way to launch the activity.	Optional
Interaction Activities	Object	Interaction definitions	Optional
extensions	Object	A map of other properties as needed	Optional

Interaction Activities (referenced in Activity Definition) Table: If implemented, the Interaction Activities Object follows this table:

Property	Type	Description	Required
interactionType	String	The type of interaction. Possible values are: true-false, choice, fill-in, long-fill-in, matching, performance, sequencing, likert, numeric or other.	Required
correctResponsesPattern	Array of Strings	A pattern representing the correct response to the interaction. The structure of this pattern varies depending on the interactionType. This is detailed below.	Optional
choices, scale, source, target, or steps	Array of interaction components	Specific to the given interactionType (see below).	Optional

Interaction Types (referenced in Interaction Activities Table): The interactionType is controlled vocabulary. Based on the term used, the data formatting and purpose changes according to the following table.

InteractionType	Description	Format
true-false	An interaction with two possible responses: true or false.	Either true or false
choice	An interaction with a number of possible choices from which the learner can select. This includes interactions in which the learner can select only one answer from the list and those where the learner can select multiple items.	A list of item ids delimited by [,]. If the response contains only one item, the delimiter *shall* not be used.
fill-in	An interaction which requires the learner to supply a short response in the form of one or more strings of characters. Typically, the correct response consists of part of a word, one word or a few words. "Short" means that the correct responses pattern and learner response strings are normally be 250 characters or less.	A list of responses delimited by [,]. If the response contains only one item, the delimiter *shall* not be used.
long-fill-in	An interaction which requires the learner to supply a response in the form of a long string of characters. "Long" means that the correct responses pattern and learner response strings are normally be more than 250 characters.	A list of responses delimited by [,]. If the response contains only one item, the delimiter *shall* not be used.
matching	An interaction where the learner is asked to match items in one set (the source set) to items in another set (the target set). Items do not have to pair off exactly and it is possible for multiple or zero source items to be matched to a given target and vice versa.	A list of matching pairs, where each pair consists of a source item id followed by a target item id. Items can appear in multiple (or zero) pairs. Items within a pair are delimited by [.]. Pairs are delimited by [,].
performance	An interaction that requires the learner to perform a task that requires multiple steps.	A list of steps containing a step ids and the response to that step. Step ids are separated from responses by [.]. Steps are delimited by [,]. The response can be a String as in a fill-in interaction or a number range as in a numeric interaction.
sequencing	An interaction where the learner is asked to order items in a set.	An ordered list of item ids delimited by [,].

Table continues

InteractionType	Description	Format
likert	An interaction which asks the learner to select from a discrete set of choices on a scale	A single item id
numeric	Any interaction which requires a numeric response from the learner.	A range of numbers represented by a minimum and a maximum delimited by [:]. Where the range does not have a maximum or does not have a minimum, that number is omitted but the delimiter is still used. E.g. [:]4 indicates a maximum for 4 and no minimum.Where the correct response or learner's response is a single number rather than a range, the single number with no delimiter *may* be used.
other	Another type of interaction that does not fit into those defined above.	Any format is valid within this string as appropriate for the type of interaction.

The Correct Responses Pattern contains an array of response patterns. A learner's response is considered correct if it matches *any* of the response patterns in that array. Where a response pattern is a delimited list, the learner's response is only considered correct if *all* of the items in that list match the learner's response.

Characterstring Parameters

Some of the values within the responses described above can be prepended with certain additional parameters. These parameters are represented by the format {parameter=value}. See the long-fill-in example in A.3.4.

Characterstring parameters are not validated by the LRS. Systems interpreting Statement data can use their best judgement in interpreting (or ignoring) invalid characterstring parameters and values.

The following parameters are valid at the start of the string representing the list of items for the listed interaction types:

Parameter	Default	Description	Value	Interaction types
case_matters	false	Whether or not the case of items in the list matters.	true or false	fill-in, long-fill-in
order_matters	true	Whether or not the order of items in the list matters.	true or false	fill-in, long-fill-in, performance

The following parameters are valid at the start of each item in the list for the listed interaction types:

Parameter	Description	Value	Interaction types
lang	The language used within the item.	RFC 5646 Language Tag	fill-in, long-fill-in, performance (String responses only)

Interaction Components, Expanded (part of the Interaction Activities Table): Depending on interactionType, Interaction Activities can take additional properties, each containing a list of interaction components. These additional properties are called "interaction component lists". The following table shows the supported interaction component list(s) for an Interaction Activity with the given interactionType.

interactionType	Interaction Component Lists	Description
choice, sequencing	choices	A list of the options available in the interaction for selection or ordering.
likert	scale	A list of the options on the likert scale.

Table continues

interactionType	Interaction Component Lists	Description
matching	source, target	Lists of sources and targets to be matched.
performance	steps	A list of the elements making up the performance interaction.
true-false, fill-in, long-fill- in, numeric, other	[No component lists supported]	None

Regardless of interactionType, each Component List has the following additional properties:

Property	Type	Description	Required
id	String	Identifies the interaction component within the list.	Required
description	Language Map	A description of the interaction component (for example, the text for a given choice in a multiple- choice interaction)	Optional

Object As Actor Table: The table below lists all properties of a Actor Object.

Property	Type	Description	Required
objectType	String	*shall* be Agent or Group when present	Required
See **Actor as Agent Table**.			

Object As Statement Table: The table below lists all properties of a Statement Reference Object.

Property	Type	Description	Required
objectType	String	In this case, *shall* be StatementRef.	Required
id	UUID	The UUID of a Statement.	Required

Object As Sub-Statement Table: The table below lists all properties of a Sub-Statement Object.

Property	Type	Description	Required
objectType	String	In this case, *shall* be SubStatement.	Required
actor	Object	Whom the Statement is about, as an Agent or Group Object.	Required
verb	Object	Action taken by the Actor.	Required
object	Object	Activity, Agent, or another Statement that is the Object of the Statement.	Required
result	Object	Result Object, further details representing a measured outcome.	Optional
context	Object	Context that gives the Statement more meaning. Examples: a team the Actor is working with, altitude at which a scenario was attempted in a flight simulator.	Optional
timestamp	Timestamp	Timestamp of when the events described within this Statement occurred (it can represent any point during an experience, not necessarily the beginning or end). Set by the LRS if not provided.	Optional
attachments	Ordered array of Attachment Objects	Headers for Attachments to the Statement	Optional

4.2.2.4 Result

Represents a measured outcome related to the Statement in which it is included.

Result Table: The table below lists all properties of the Result Object.

Property	Type	Description	Required
score	Object	The score of the Agent in relation to the success or quality of the experience.	Optional
success	Boolean	Indicates whether or not the attempt on the Activity was successful.	Optional
completion	Boolean	Indicates whether or not the Activity was completed.	Optional
response	String	A response appropriately formatted for the given Activity.	Optional
duration	Duration	Period of time over which the Statement occurred.	Optional
extensions	Object	A map of other properties as needed.	Optional

Score Table: Represents the outcome of a graded Activity achieved by an Agent. The table below lists all properties of the Score Object.

Property	Type	Description	Required
scaled	Decimal number between −1 and 1, inclusive	The score related to the experience as modified by scaling and/or normalization.	Recommended
raw	Decimal number between min and max (if present, otherwise unrestricted), inclusive	The score achieved by the Actor in the experience described by the Statement. This is not modified by any scaling or normalization.	Optional
min	Decimal number less than max (if present)	The lowest possible score for the experience described by the Statement.	Optional
max	Decimal number greater than min (if present)	The highest possible score for the experience described by the Statement.	Optional

4.2.2.5 Context

Property to add contextual information to a Statement. It can store information such as the instructor for an experience, if this experience happened as part of a team-based Activity, or how an experience fits into some broader activity.

Context Table: The table below lists all properties of the Context Object.

Property	Type	Description	Required
registration	UUID	The registration that the Statement is associated with.	Optional
instructor	Agent (*may* be a Group)	Instructor that the Statement relates to, if not included as the Actor of the Statement.	Not Recommended
team	Group	Team that this Statement relates to, if not included as the Actor of the Statement.	Not Recommended
contextActivities	contextActivities Object	A map of the types of learning activity context that this Statement is related to. Every key in the contextActivities Object *shall* be one of "parent", "grouping", "category", or "other". Every value in the contextActivities Object *shall* be either a single Activity Object or an array of Activity Objects.	Optional

Property	Type	Description	Required
contextAgents	Array of contextAgent Objects	Collection of Objects describing relationship(s) between Agent(s) and the current Statement. Zero or more Relevant Type IRIs are used to categorize these relationship(s)	Optional
contextGroups	Array of contextGroup Objects	Collection of Objects describing relationship(s) between Identified or Anonymous Group(s) and the current Statement. Zero or more Relevant Type IRIs are used to categorize these relationship(s)	Optional
revision	String	Revision of the learning activity associated with this Statement. Format is free.	Optional
platform	String	Platform used in the experience of this learning activity.	Optional
language	String (as defined in RFC 5646)	Code representing the language in which the experience being recorded in this Statement (mainly) occurred in, if applicable and known.	Optional
statement	Statement Reference	Another Statement to be considered as context for this Statement.	Optional
extensions	Object	A map of any other domain-specific context relevant to this Statement. For example, in a flight simulator altitude, airspeed, wind, attitude, GPS coordinates might all be relevant	Optional

NOTE—With the addition of context agents and context groups, it is recommended not to use instructor or team. They are supported in this document for backward compatibility purposes.

Context Agents and Groups Details:

A single Statement may require the inclusion of many contextually relevant Agent(s) and/or Group(s) in order to properly describe an experience. When this is the case, the relationship between the Agent(s) and/or Group(s) and the Statement itself needs to be represented in a structured manner. The context properties contextAgents and contextGroups serve as this structure.

— Inclusion of contextAgents within a Statement establishes that a relationship exists between said Statement and one or more Agent(s)

— Inclusion of contextGroups within a Statement establishes that a relationship exists between said Statement and one or more Group(s)

— Zero or more Relevant Types are used to categorize each Statement specific relationship

— Each Statement specific relationship corresponds to an individual contextAgents or contextGroups Object

The relationship established by each contextAgents and contextGroups Object is limited in scope to the Statement in which the Object is found. In general, an Agent many have permanent characteristics, characteristics which are consistent across experiences, but these kinds of Agent specific properties should be captured in an Agent Profile

All Objects found within the contextAgents and/or contextGroups array(s) are independent of one another.

Context Agents Table: Each contextAgent Object found within a contextAgents array has the following properties.

Property	Type	Description	Required
objectType	String	contextAgent	Required
agent	Agent Object	A single Agent Object for which a Statement specific relationship is being defined	Required
relevantTypes	Array of Relevant "type" IRIs	A collection of 1 or more Relevant Type(s) used to characterize the relationship between the Statement and the Agent. If not provided, only a generic relationship is intended (not recommended)	Optional

— Any and All valid Agent Objects can be used as the agent within a contextAgent Object.

— An Agent Object does *not* need to be found within any other Statement property in order to be included as the agent property within a contextAgent Object.

— Any and All valid Relevant Type IRIs can be included within the relevantTypes array of a contextAgent Object.

— Any Relevant Type IRI does *not* need to be found within any other Statement property in order to be included within the relevantTypes property of a contextAgent Object.

Context Group Table: Each contextGroup Object found within a contextGroups array has the following properties.

Property	Type	Description	Required
objectType	String	contextGroup	Required
group	Group	A single Group Object for which a Statement specific relationship is being defined.	Required
relevantTypes	Array of Relevant "type" IRIs	A collection of 1 or more Relevant Type(s) used to characterize the relationship between the Statement and the Agent. If not provided, only a generic relationship is intended (not recommended)	Optional

— Any and All valid Group Objects can be used as the group within a contextGroup Object.

— A Group Object does *not* need to be found within any other Statement property in order to be included as the group property within a contextGroup Object.

— Any and All valid Relevant Type IRIs can be included within the relevantTypes array of a contextAgent Object.

— Any Relevant Type IRI does *not* need to be found within any other Statement property in order to be included within the relevantTypes property of a contextAgent Object.

4.2.2.6 Attachments

In some cases an Attachment is logically an important part of a Learning Record. It could be an essay, a video, etc. Another example of such an Attachment is (the image of) a certificate that was granted as a result of an experience.

Attachments Table: The table below lists all properties of the Attachments Object.

Property	Type	Description	Required	Corresponding Request Parameter
usageType	IRI	Identifies the usage of this Attachment. For example: one expected use case for Attachments is to include a "completion certificate". An IRI corresponding to this usage *shall* be coined, and used with completion certificate attachments.	Required	
display	Language Map	Display name (title) of this Attachment.	Required	
description	Language Map	A description of the Attachment	Optional	
contentType	Internet Media Type	The content type of the Attachment.	Required	Content-Type
length	Integer	The length of the Attachment data in octets.	Required	Content-Length
sha2	String	The SHA-2 (FIPS PUB 180-2) hash of the Attachment data. This property is always required, even if fileUrl is also specified.	Required	X-Experience-API-Hash
fileUrl	IRL	An IRL at which the Attachment data can be retrieved, or from which it used to be retrievable.	Optional	

4.2.3 Metadata

Metadata is additional information about a resource. It enables decision making, search, and discoverability. In xAPI, metadata can be utilized essentially in two ways.

First, metadata is used in the definition property of an Activity Object (the Object when the objectType is Activity). In this case, the metadata is part of the Statement. Other fields, including extensions, can be used as metadata in a similar regard. Including metadata in a Statement allows metadata about the IRI to be expressed without the necessity of resolving it.

Second, metadata may be hosted. In this case, additional information about an identifier can be provided within a Statement and can be hosted at the location pointed to by the identifier IRI. Hosting metadata at the IRI location allows the owner of the IRI to define the canonical metadata for that IRI. This is the main purpose for which IRIs are used in xAPI.

All hosted metadata, including the format, is optional. However, it is recommended that the Activity Definition Object metadata is followed for hosted Activity identifiers.

For the structure of metadata about all other identifiers, see the format below:

Property	Type	Description	Required
name	Language Map	The human readable/visual name. For Verbs, this is equivalent to the "display" property in a Statement.	Optional
description	Language Map	Description	Optional

Hosted metadata consists of a document containing a JSON object as described above. If this hosted metadata is provided, it is the canonical source of information about the identifier it describes.

4.2.3.1 LRS Recommendations

The following recommendations are made for any LRS that implements resolvable metadata into its data model for querying purposes.

— If an Activity IRI is a URL, an LRS *should* attempt to GET that URL, and include in HTTP headers: Accept: application/json, */*. This *should* be done as soon as practical after the LRS first encounters the Activity id.

— Upon loading JSON which is a valid Activity Definition from a URL used as an Activity id, an LRS *should* incorporate the loaded definition into its canonical definition for that Activity, while preserving names or definitions not included in the loaded definition.

— Upon loading any document from which the LRS can parse an Activity Definition from a URL used as an Activity id, an LRS *may* consider this definition when determining its canonical representation of that Activity's definition.

4.2.4 LRS Processing of Data

An LRS functions as a gatekeeper for xAPI Data. Statements, in particular, are highly structured and have many requirements. Statements are expected to be unique and to be immutable. In this regard they are unchangeable and should not be deleted. This subclause outlines requirements that focus on storage and retrieval of Statements, including the rejection cases for Statements.

4.2.4.1 LRS Rejection Cases

It is useful to note that an LRS can reject Statements for many reasons not specified in this document and does not have to perpetually keep data beyond normal processes. This specification is not intended to usurp security, authority, or data management processes.

The following requirements have to do with rejection/acceptance in such cases:

— The LRS *should* reject entire batches of Statements if sent from an unauthorized source

— An LRS *may not* reject a Statement solely based on the order of properties

— The LRS shall not reject a Statement that uses the voided verb if it cannot find the id of the Object of that Statement (nor does the LRS have to try to find it)

— An LRS *should not* reject a Statement based on size of individual properties

— An LRS *may* reject a Statement if the overall size is too large, contains information not permissible to the environment, or is thought to be malicious

— An LRS *may* choose to not validate IRIs/UUIDs; An LRS is responsible for data format, not values

— An LRS shall *not* reject a timestamp for having a greater value than the current time, within an acceptable margin of error (intentionally not specified in this document)

4.2.4.2 Specific Statement Data Requirements for an LRS

The following requirements apply to specific parts of a component. These requirements involve rejection, creation, conformance to standards, and overwriting of data.

— The LRS *shall not* return a different serialization of any properties except those as described in 4.2.

— The LRS *shall* create an "id" property if an accepted Statement does not have one.

— The LRS shall reject Statements if the Actor object contains more than one Inverse Functional Identifier.

— The LRS shall reject Statements which use an Agent of type "Group" within the member property of a Group.

— The LRS *shall* process and store numbers with at least the precision of IEEE 754 32-bit floating point numbers.

— The LRS shall reject a Statement that uses an Interaction Activity without a valid interactionType.

— An LRS, upon consuming a valid interactionType, *may* validate the remaining properties as specified for Interaction Activities and *may* return 400 Bad Request if the remaining properties are not valid for the Interaction Activity.

— An interaction component's id value *should not* have whitespace.

— The LRS shall reject a Statement with an Object with ObjectType SubStatement that has within it an Object with ObjectType SubStatement.

— The LRS shall set the "timestamp" property to the value of the "stored" property if not provided.

— The "stored" property *shall* be set by the LRS; An LRS *should* validate and then *shall* overwrite any value currently in the "stored" property of a Statement it receives.

— The LRS *shall* ensure that all Statements stored have an authority.

— The LRS *should* overwrite the authority on all Statements it stores, based on the credentials used to send those Statements.

— When the LRS overwrites the authority, the LRS *shall* apply a deterministic process to map the credentials used to store a statement to an authority.

— The LRS *may* leave the submitted authority unchanged but *should* do so only where a strong trust relationship has been established, and with extreme caution.

— The LRS *shall* return every value in the contextActivities Object as an array, even if it arrived as a single Activity Object.

— The LRS *shall* return single Activity Objects as an array of length one containing the same Activity.

— An LRS *should not* reject a Statement that is otherwise valid except the version (property within Statement).

4.2.4.3 Data Response Recommendations

The following optional recommendations are determined best practices to some of the fringe components of xAPI. This subclause can be used to detail processes that come up in exceptional cases.

Upon receiving a Statement with an Activity Definition that differs from the one stored, an LRS *should* decide whether it considers the Learning Record Provider to have the authority to change the definition and *should* update the stored Activity Definition accordingly if that decision is positive.

An LRS *may* make small corrections to its canonical definition for the Activity when receiving a new definition e.g., spelling error.

An LRS *should not* make significant changes to its canonical definition for the Activity based on an updated definition e.g., changes to correct responses.

Statements returned by an LRS *shall* retain the version they are accepted with. If they lack a version, the version *shall* be set to 2.0.0.

An LRS *may* include all necessary information within the "more" property IRL to continue the query to avoid the need to store IRLs and associated query data.

An LRS *should not* generate extremely long IRLs within the "more" property.

An LRS *may* re-run the query at the point in time that the IRL retrieved from the "more" property is accessed such that the batch retrieved includes Statements which would have been included in that batch if present in the LRS at the time the original query was run and excludes Statements from that batch which have since been voided.

Alternatively, an LRS *may* cache a list of Statements to be returned at the "more" property such that the batch of Statements returned matches those Statements that would have been returned when the original query was run.

An LRS *may* remove voided Statements from the cached list of Statements if using this method.

4.2.5 Statement Voiding

Not all Statements are perpetually valid once they have been issued. Mistakes or other factors could dictate that a previously made Statement is marked as invalid. This is called "voiding a Statement" and the reserved Verb "http://adlnet.gov/expapi/verbs/voided" is used for this purpose. Any Statement that voids another cannot itself be voided.

The LRS shall reject Statements with the verb http://adlnet.gov/expapi/verbs/voided when:

- The objectType property is not "StatementRef"
- Has no "id" for the Object (e.g., no target for voiding)

An LRS *shall* consider a Statement it contains voided if and only if the Statement is not itself a voiding Statement and the LRS also contains a voiding Statement referring to the first Statement.

Upon receiving a Statement that voids another, the LRS *may* roll back any changes to Activity or Agent definitions which were introduced by the Statement that was just voided.

4.2.6 Statement Signing

A Statement can include a digital signature to provide strong and durable evidence of the authenticity and integrity of the Statement.

Signed Statements include a JSON web signature (JWS) as an Attachment. This allows the original serialization of the Statement to be included along with the signature. For interoperability, the "RSA + SHA" series of JWS algorithms have been selected, and for discoverability of the signer X.509 certificates *should* be used.

Statement Signing Process:

- A Signed Statement *shall* include a JSON web signature (JWS) as defined in RFC 7515, as an Attachment with a usageType of "http://adlnet.gov/expapi/attachments/signature" and a contentType of application/octet-stream.
- JWS Compact Serialization *shall* be used to create the JSON web signature. Use of JWS JSON Serialization is strongly discouraged, is unlikely to be interoperable with other systems.
- The JWS signature *shall* have a payload of a valid JSON serialization of the complete Statement before the signature was added.

— The JWS signature *shall* use an algorithm of "RS256", "RS384", or "RS512".

— The JWS signature *should* have been created based on the private key associated with an X.509 certificate.

— If X.509 was used to sign, the JWS header *should* include the "x5c" property containing the associated certificate chain.

Additional Requirements

— The LRS *shall* reject requests to store Statements that contain malformed signatures, with 400 Bad Request.

— The LRS *should* include a message in the response of a rejected statement.

— In order to verify signatures are well formed, the LRS *shall* do the following:

  — Decode the JWS signature and load the signed serialization of the Statement from the JWS signature payload.

  — Validate that the original Statement is logically equivalent to the received Statement.

  — If the JWS header includes an X.509 certificate, validate the signature against that certificate as defined in JWS.

  — Validate that the signature requirements outlined above have been met.

4.2.7 Additional Requirements for Data Types

The following subclauses provide guidance and requirements for data types found in this document. Many tables contain specific data types that have requirements found in this subclause.

4.2.7.1 IRIs

Internationalized Resource Identifiers, or IRIs, are unique identifiers which could also be resolvable. Because resolving is not a requirement, IRIs/Uniform Resource Identifiers (URIs) are used instead of IRLs/URLs. In order to allow the greatest flexibility in the characters used in an identifier, IRIs are used instead of URIs as IRIs can contain some characters outside of the ASCII character set.

The LRS has responsibilities in regard to each IRI as outlined as follows:

— When storing or comparing IRIs, LRSs *shall* handle them only by using one or more of the approaches described in 5.3.1 (Simple String Comparison) and 5.3.2 (Syntax-Based Normalization) of RFC 3987

4.2.7.2 Extensions

Extensions are available as part of Activity Definitions, as part of a Statement's "context" property, or as part of a Statement's "result" property. In each case, extensions are intended to provide a natural way to extend those properties for some specialized use. The contents of these extensions might be something valuable to just one application, or it might be a convention used by an entire Community of Practice.

Extensions are defined by a map and logically relate to the part of the Statement where they are present. The values of an extension can be any JSON value or data structure. Extensions in the "context" property provide context to the core experience, while those in the "result" property provide elements related to some outcome. Within Activities, extensions provide additional information that helps define an Activity within some custom application or Community of Practice. The meaning and structure of extension values under an IRI key are defined by the person who controls the IRI.

— The LRS *shall* reject any Statement where a key of an extensions map is not an IRI.

— An LRS *shall not* reject a Statement based on the values of the extensions map.

4.2.7.3 Language Maps

A language map is a dictionary where the key is an RFC 5646 Language Tag, and the value is a string in the language specified in the tag. This map *should* be populated as fully as possible based on the knowledge of the string in question in different languages.

The shortest valid language code for each language string is generally preferred. The ISO 639 language code plus an ISO 3166-1 country code allows for the designation of basic languages (e.g., es for Spanish) and regions (e.g., es-MX, the dialect of Spanish spoken in Mexico). If only the ISO 639 language code is known for certain, do not guess at the possible ISO 3166-1 country code. For example, if only the primary language is known (e.g., English) then use the top-level language tag en, rather than en-US. If the specific regional variation is known, then use the full language code.

NOTE—For Chinese languages, the significant linguistic diversity represented by "zh" means that the ISO 639 language code is generally insufficient.

The content of strings within a language map is plain text. It is expected that any formatting code such as HTML tags or markdown is not rendered but displayed as code when this string is displayed to an end user. An important exception to this is if language map Object is used in an extension and the owner of that extension IRI explicitly states that a particular form of code is to be rendered.

— An LRS *shall* reject a Statement with invalid RFC 5646 language tags (Keys of language maps *shall* be sent with valid RFC 5646 language tags, for similar reasons

— The LRS *shall* at least validate that the sequence of token lengths for language map keys matches the RFC 5646 standard

4.2.7.4 UUIDs

Universally Unique Identifiers, or UUIDs, are 128-bit values that are globally unique. Unlike IRIs, there is no expectation of resolvability as UUIDs take on a completely different format.

— UUIDs *shall* be in the standard string form.

— UUIDS *should* use variant 2 in RFC 4122.

4.2.7.5 Timestamps

— An LRS *shall* convert Timestamps to UTC rather than rejecting Statements that send Timestamps not in UTC form

— An LRS *may* truncated or round a Timestamp to a precision of at least 3 decimal digits for seconds.

4.2.7.6 Duration

— The LRS *shall* reject a Statement if the duration is not expressed using the format for Duration in ISO 8601:2004(E) section 4.4.3.2.

— An LRS *shall* reject Statements with the alternative format (in conformity with the format used for time points and described in ISO 8601:2004(E) section 4.4.3.3). This requirement exists in case of conflicting ISO requirements.

— On receiving a Duration with more than 0.01 s precision, the LRS *shall not* reject the request but *may* truncate the "duration" property to 0.01 s precision.

— When making a comparison (e.g., as a part of the statement signing process) of Statements in regard to a Duration, any precision beyond 0.01 s precision *shall not* be included in the comparison.

4.2.7.7 Documents

Note that the following table shows generic properties, not a JSON Object as many other tables in this specification do. The id is stored in the IRL, "updated" is HTTP header information, and "contents" is the HTTP document itself (as opposed to an Object).

Property	Type	Description
id	String	Set by Learning Record Provider, unique within the scope of the Agent or Activity.
updated	Timestamp	When the document was most recently modified (HTTP Header).
contents	Arbitrary binary data	The contents of the document

5. Content—Learning Record Providers (LRPs) and Learning Record Consumers (LRCs)

5.1 LRP and LRC Communication

LRPs and LRCs interact with an LRS via RESTful HTTP methods to the resources outlined in this subclause. The primary function of LRPs and LRCs within the xAPI is to create valid Requests to the LRS. All tables in this subclause are redundant with those listed in 4.1, but are listed here as context such that LRPs can understand what the LRS supports. This subclause does include LRP requirements that are not seen in 4.1.

Examples of the resources and data requirements can be found at https://opensource.ieee.org/xapi/xapi-base-standard-examples.

The following table summarizes the Resources with which HTTP methods can interact. Details for what an LRP can expect can be found within 4.1.6.

Base Resource IRI/URL of the LRS Precedes Each Entry	Function	Supported Calls
statements	Statement Storage/Retrieval	PUT, POST, GET,HEAD
agents	Agent Object Storage/Retrieval	GET, HEAD
agents/profile	Agent Profile Resource	PUT, POST, GET, HEAD, DELETE
activities	Activity Object Storage/Retrieval	GET, HEAD
activities/profile	Activity Profile Resource	PUT, POST, GET, HEAD, DELETE
activities/state	State Resource	PUT, POST, GET, DELETE, HEAD
about	LRS Information	GET, HEAD
extensions/"yourext"	Any Additional Resource Not Identified in this Document	Not specified

5.1.1 Headers

The LRP is allowed to use the following headers as described in this document.

- Accept
- Accept-Encoding
- Accept-Language
- Authorization
- Content-Type
- Content-Length
- Content-Transfer-Encoding
- If-Match
- If-None-Match
- X-Experience-API-Version

5.1.2 Encoding

All strings *shall* be encoded and interpreted as UTF-8 (RFC 3629).

5.1.3 Content Types

Requests and responses within this specification normally use an application/json content type. Exceptions to this are:

- Documents can have any content type.
- Statement requests that can sometimes include Attachments use the multipart/mixed content type.

5.1.3.1 Application/JSON

The application/json content type is used for all requests that do not otherwise specify a content type.

5.1.3.2 Multipart/Mixed

The multipart/mixed content type is used for requests that *could* include Attachments. This does not mean that all "multipart/mixed" requests necessarily do include Attachments.

5.1.3.3 Procedure For The Exchange Of Attachments

- A Statement request including zero or more Attachments is construed in accordance with requirements in this document.
- The Statement is sent using a Content-Type of multipart/mixed. Any Attachments are placed at the end of such transmissions.
- The LRS decides whether to accept or reject the Statement based on the information in the first part.
- If it accepts the request, it can match the raw data of an Attachment(s) with the Attachment header by comparing the SHA-2 (FIPS PUB 180-2) of the raw data to the SHA-2 declared in the header. It *shall* not do so any other way.

5.1.3.4 Requirements for Attachment Statement Batches

A request transmitting a Statement batch, Statement results, or single Statement that includes Attachments *shall* satisfy one of the following criteria:

- It *shall* be of type application/json and include a fileUrl for every Attachment EXCEPT for Statement results when the "attachments" filter is false.
- It *shall* conform to the definition of "multipart/mixed" in RFC 2046 and:
  - The first part of the multipart document *shall* contain the Statements themselves, with type application/json.
  - Each additional part contains the raw data for an Attachment and forms a logical part of the Statement. This capability *shall* be available when issuing PUT or POST requests against the Statement Resource.
  - *shall* include an X-Experience-API-Hash parameter in each part's header after the first (Statements) part.
  - *shall* include a Content-Transfer-Encoding parameter with a value of binary in each part's header after the first (Statements) part.
  - *should* only include one copy of an Attachment's data when the same Attachment is used in multiple Statements that are sent together.
  - *should* include a Content-Type parameter in each part's header. For the first part (containing the Statement) this *shall* be application/json.
  - Where parameters have a corresponding property within the attachment Object and both the parameter and property are specified for a given Attachment, the value of these parameters and properties *shall* match.

5.1.3.5 Learning Record Provider Requirements

- A Learning Record Provider *may* send Statements with Attachments as described above.
- A Learning Record Provider *may* send multiple Statements where some or all have Attachments if using POST.
- A Learning Record Provider *may* send batches of type application/json where every attachment Object has a fileUrl, ignoring all requirements based on the "multipart/mixed" format.
- A Learning Record Provider *should* use SHA-256, SHA-384, or SHA-512 (FIPS PUB 180-2) to populate the "sha2" property.

5.1.3.6 File URL

The File URL is intended to provide a location from which the Attachment can be received. There are, however, no requirements for the owner of the Attachment to make the Attachment data available at the location indefinitely or to make the Attachment publicly available without security restrictions.

When determining Attachment hosting arrangements, those creating Statements using the "fileUrl" property are encouraged to consider the needs of end recipient(s) of the Statement, especially if the Attachment content is not included with the Statement.

The period of time an Attachment is made available for, and the security restrictions applied to hosted attachments, are out of scope of this specification.

5.1.4 Concurrency

Concurrency control makes certain that a PUT, POST or DELETE does not perform operations based on old data.

5.1.4.1 Details

xAPI uses HTTP 1.1 entity tags (ETags) to implement optimistic concurrency control in the following resources, where PUT, POST or DELETE are allowed to overwrite or remove existing data:

— State Resource

— Agent Profile Resource

— Activity Profile Resource

5.1.4.2 Requirements

— As LRP making a PUT request to the State Resource, Agent Profile Resource or Activity Profile Resource *shall* include the "If-Match" header or the If-None-Match header.

— An LRP making a POST request to the State Resource, Agent Profile Resource or Activity Profile Resource *shall* include the "If-Match" header or the If-None-Match header.

— An LRP making a DELETE request to the State Resource, Agent Profile Resource or Activity Profile Resource *shall* include the "If-Match" header.

— An LRP *shall* use the ETag value provided by the LRS rather than calculating it themselves.

5.1.5 Error Codes

The list below covers many error conditions that could be returned from various methods in the API.

— 400 Bad Request: Indicates an error condition caused by an invalid or missing argument. The term "invalid arguments" includes malformed JSON (RFC 8259) or invalid Object structures.

— 401 Unauthorized: Indicates that authentication is required, or in the case authentication has been posted in the request, that the given credentials have been refused.

— 403 Forbidden: Indicates that the request is unauthorized for the given credentials. Note this is different than refusing the credentials given. In this case, the credentials have been validated, but the authenticated system is not allowed to perform the given action.

— 404 Not Found: Indicates the requested resource was not found. May be returned by any method that returns a uniquely identified resource, for instance, any State, Agent Profile, or Activity Profile Resource request targeting a specific document, or the method to retrieve a single Statement.

— 409 Conflict: Indicates an error condition due to a conflict with the current state of a resource, in the case of State Resource, Agent Profile Resource or Activity Profile Resource requests, or in the Statement Resource PUT or POST calls.

— 412 Precondition Failed: Indicates an error condition due to a failure of a precondition posted with the request, in the case of State or Agent Profile or Activity Profile API requests.

— 413 Request Entity Too Large: Indicates that the LRS has rejected the Statement or document because its size (or the size of an Attachment included in the request) is larger than the maximum allowed by the LRS.

— 429 Too Many Requests: Indicates that the LRS has rejected the request because it has received too many requests from the system or set of credentials in a given amount of time.

— 500 Internal Server Error: Indicates a general error condition, typically an unexpected exception in processing on the server.

Requirements

A Learning Record Provider *should* send an "Accept" header with requests to enable content negotiation.

5.1.6 LRS Resources

Each LRS Resource is described in 5.1.6. Each Resource has requirements for each of the HTTP methods that may be made to it. Each subclause includes the expected contents of the body of the request, the request parameters, and the expected returns. If an HTTP method is not described, it is out of scope of this document.

LRPs leverage both Statements and documents as data structures. An LRP may interact with any of the resources described in this subclause.

A Learning Record Provider *may* send documents to any of the Document Resources for Activities and Agents that the LRS does not have prior knowledge of.

For Groups, it is only necessary to provide enough information for the LRS to construct the Inverse Functional Identifier (IFI) of the Agent object. In the case of an Identified Group, the array of members does not contribute to the IFI and may be omitted (If the array of members is included, the LRS will ignore the members and only use the IFI for comparison purposes).

NOTE—In all of the examples in this specification, http://example.com/xAPI/ is the example base endpoint (resource location) of the LRS. All other IRI syntax after this represents the particular resource used.

Examples of the resources can be found at https://opensource.ieee.org/xapi/xapi-base-standard-examples.

5.1.6.1 Statement Resource (/statements)

Statements are the key data structure of xAPI. This resource facilitates their storage and retrieval.

Example resource location/endpoint: http://example.com/xAPI/statements

5.1.6.1.1 PUT Request

Summary: Stores a single Statement with the given id.

Parameter	Type	Default	Description	Required
statementId	String		Id of Statement to record	Required

Body: The Statement object to be stored.

Returns: 204 No Content

Learning Record Provider Requirements

— Learning Record Providers *should* POST Statements including the Statement "id" property instead of using PUT.

— When PUTing Statements, the "id" property of the Statement *should* be used.

— Where provided, the "id" property of the Statement *shall* match the "statementId" parameter of the request.

5.1.6.1.2 POST Request

Summary: Stores a Statement, or a set of Statements.

Body: An array of Statements or a single Statement to be stored.

Returns: 200 OK, Array of Statement id(s) (UUID) in the same order as the corresponding stored Statements

5.1.6.1.3 GET Request

Summary: Fetches a single Statement or multiple Statements. If the statementId or voidedStatementId parameter is specified a single Statement is returned. Otherwise, a StatementResult Object, a list of Statements in reverse chronological order based on "stored" time, subject to permissions and maximum list length. If additional results are available, an IRL to retrieve them is included in the StatementResult Object.

Parameter	Type	Default	Description	Required
statementId	String		Id of Statement to fetch	Optional
voidedStatementId	String		Id of voided Statement to fetch. see Voided Statements	Optional
agent	Agent or Identified Group Object (JSON)		Filter, only return Statements for which the specified Agent or Group is the Actor or Object of the Statement. • Agents or Identified Groups are equal when the same Inverse Functional Identifier is used in each Object compared and those Inverse Functional Identifiers have equal values. • For the purposes of this filter, Groups that have members which match the specified Agent based on their Inverse Functional Identifier as described above are considered a match. See agent/group Object definition for details.	Optional
verb	Verb id (IRI)		Filter, only return Statements matching the specified Verb id.	Optional
activity	Activity id (IRI)		Filter, only return Statements for which the Object of the Statement is an Activity with the specified id.	Optional

Table continues

Parameter	Type	Default	Description	Required
registration	UUID		Filter, only return Statements matching the specified registration id. Note that although frequently a unique registration is used for one Actor assigned to one Activity, this cannot be assumed. If only Statements for a certain Actor or Activity are required, those parameters also need to be specified.	Optional
related_activities	Boolean	false	Apply the Activity filter broadly. Include Statements for which the Object, any of the context Activities, or any of those properties in a contained SubStatement match the Activity parameter, instead of that parameter's normal behavior. Matching is defined in the same way it is for the "activity" parameter.	Optional
related_agents	Boolean	false	Apply the Agent filter broadly. Include Statements for which the Actor, Object, Authority, Instructor, Team,	Optional
			Context Agent, Context Group, or any of these properties in a contained SubStatement match the Agent parameter, instead of that parameter's normal behavior. Matching is defined in the same way it is for the "agent" parameter.	
since	Timestamp		Only Statements stored since the specified Timestamp (exclusive) are returned.	Optional
until	Timestamp		Only Statements stored at or before the specified Timestamp are returned.	Optional
limit	Nonnegative Integer	0	Maximum number of Statements to return. 0 indicates return the maximum the server allows.	Optional

Table continues

Parameter	Type	Default	Description	Required
format	String: (ids, exact, or canonical)	exact	If ids, only include minimum information necessary in Agent, Activity, Verb and Group Objects to identify them. For Anonymous Groups this means including the minimum information needed to identify each member. If exact, return Agent, Activity, Verb and Group Objects populated exactly as they were when the Statement was received. An LRS requesting Statements for the purpose of importing them would use a format of "exact" in order to maintain Statement Immutability. If canonical, return Activity Objects and Verbs populated with the canonical definition of the Activity Objects and Display of the Verbs as determined by the LRS, after applying the language filtering process defined below, and return the original Agent and Group Objects as in "exact" mode.	Optional
attachments	Boolean	false	If true, the LRS uses the multipart response format and includes all attachments as described previously. If false, the LRS sends the prescribed response with Content-Type application/json and does not send attachment data.	Optional
ascending	Boolean	false	If true, return results in ascending order of stored time	Optional

Body: None.

Returns: 200 OK, Statement or Statement Result

NOTE—The values of Boolean parameters are represented as true or false as in JSON.

Voided Statements

This subclause describes how voided Statements are handled by the LRS when queried via a GET request.

Learning Record Providers can identify the presence and Statement id of any voided Statements by the target of the voiding Statement.

5.1.6.2 State Resource (/activities/state)

Generally, this is a scratch area for Learning Record Providers that do not have their own internal storage, or need to persist state across devices. The State Resource is a place to store information about the state of an activity in a generic form called a document. The intent of this resource is to store/retrieve a specific Agent or Identified Group's data within a specific activity, potentially tied to a registration.

The semantics of the LRS response are driven by the presence of a "stateId" parameter. If it is included, the GET and DELETE methods *shall* act upon a single defined state document identified by "stateId". Otherwise, GET returns the available ids, and DELETE deletes all state in the context given through the other parameters. This resource has concurrency controls associated with it.

Example resource location/endpoint: http://example.com/xAPI/activities/state

5.1.6.2.1 PUT Request

Summary: Stores a single document with the given id.

Body: The document object to be stored.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this state.	Required
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required

5.1.6.2.2 POST Request

Summary: Updates/stores a single document with the given id.

Body: The document object to be stored/updated.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this state.	Required
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required

If a Learning Record Provider needs to delete a property, it *should* use a PUT request to replace the whole document.

5.1.6.2.3 (Single Document) GET Request

Summary: Fetches a single document with the given id.

Body: None.

Returns: 200, The State Document

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this state.	Required
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required unless since parameter is present. In that case, Not Allowed.
since	Timestamp	Do not use for single document GET request.	Optional if stateId is not used. If the stateId parameter is included, Not Allowed.

5.1.6.2.4 (Single Document) DELETE Request

Summary: Deletes a single document with the given id.

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this state.	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this state.	Required
registration	UUID	The registration associated with this state.	Optional
stateId	String	The id for this state, within the given context.	Required

5.1.6.2.5 (Multiple Document) GET Request

Summary: Fetches State ids of all state data for this context (Activity + Agent (or Identified Group) [+ registration if specified]). If "since" parameter is specified, this is limited to entries that have been stored or updated since the specified timestamp (exclusive).

Body: None.

Returns: 200, Array of State ids

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with these states.	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with these states.	Required
registration	UUID	The Registration associated with these states.	Optional
stateId	String	Do not use for Multiple Document GET request.	Required unless since parameter is present. In that case, Not Allowed.
since	Timestamp	Only ids of states stored since the specified Timestamp (exclusive) are returned.	Optional as long as stateId is not used. If the stateId parameter is included, Not Allowed.

5.1.6.2.6 (Multiple Document) DELETE Request

Summary: Deletes all state data for this context (Activity + Agent [+ registration if specified]).

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with these states.	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with these states.	Required
registration	UUID	The Registration associated with these states.	Optional

5.1.6.3 Agents Resource (/agents)

The Agents Resource provides a method to retrieve a special Object with combined information about an Agent derived from an outside service, such as a directory service. This object is called a "Person Object". The Person Object is very similar to an Agent Object, but instead of each attribute having a single value, each attribute has an array value, and it is legal to include multiple identifying properties. This is different from the FOAF concept of person, person is being used here to indicate a person- centric view of the LRS Agent data, but Agents just refer to one persona (a person in one context).

Example resource location/endpoint: http://example.com/xAPI/agents

5.1.6.3.1 GET Request

Summary: Return a Person Object for a specified Agent.

Body: None.

Returns: 200 OK, Person Object

Parameter	Type	Description	Required
agent	Agent object (JSON)	The Agent representation to use in fetching expanded Agent information.	Required

Person Object Table:

Property	Type	Description	Required
objectType	String	Person	Required
name	Array of strings.	List of names of Agents retrieved.	Optional
mbox	Array of IRIs in the form "mailto:email address".	List of e-mail addresses of Agents retrieved.	Optional
mbox_sha1sum	Array of strings.	List of the SHA1 hashes of mailto IRIs (such as go in an mbox property).	Optional
openid	Array of strings.	List of openids that uniquely identify the Agents retrieved.	Optional
account	Array of account objects.	List of accounts to match. Complete account Objects (homePage and name) *shall* be provided.	Optional

All array properties *shall* be populated with members with the same definition as the similarly named property from Agent Objects.

5.1.6.4 Activities Resource (/activities)

The Activities Resource provides a method to retrieve a full description of an Activity from the LRS.

Example resource location/endpoint: http://example.com/xAPI/activities

5.1.6.4.1 GET Request

Summary: Fetches the complete Activity Object specified.

Body: None.

Returns: 200 OK, Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The id associated with the Activities to load.	Required

5.1.6.5 Agent Profile Resource (/agents/profile)

A place to store information about an Agent or Identified Group in a generic form called a document. This information is not tied to an activity or registration. The semantics of the LRS response are driven by the presence of a "profileId" parameter. If it is included, the GET and method acts upon a single defined profile document identified by "profileId". Otherwise, GET returns the available ids given through the other parameter. This resource has concurrency controls associated with it.

Example resource location/endpoint: http://example.com/xAPI/agents/profile

5.1.6.5.1 PUT Request

Summary: Stores a single document with the given id.

Body: The document to be stored.

Returns: 204 No Content

Parameter	Type	Description	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

An LRP making a PUT request to this resource *shall* include the "If-Match" header or the If-None-Match header.

5.1.6.5.2 POST Request

Summary: Stores/updates a single document with the given id.

Body: The document to be stored/updated.

Returns: 204 No Content

Parameter	Type	Description	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

— If a Learning Record Provider needs to delete a property, it *should* use a PUT request to replace the whole document.

— An LRP making a POST request to this resource *shall* include the "If-Match" header or the If- None-Match header.

5.1.6.5.3 DELETE Request

Summary: Deletes a single document with the given id.

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

An LRP making a DELETE request to this resource *shall* include the "If-Match" header.

5.1.6.5.4 (Single Document) GET Request

Summary: Fetches a single document with the given id.

Body: None.

Returns: 200 OK, the Profile document

Parameter	Type	Description	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

5.1.6.5.5 (Multiple Document) GET Request

Summary: Fetches Profile ids of all Profile documents for an Agent or Identified Group. If "since" parameter is specified, this is limited to entries that have been stored or updated since the specified Timestamp (exclusive).

Body: None.

Returns: 200 OK, Array of profileIds.

Parameter	Type	Description	Required
agent	Agent or Identified Group Object (JSON)	The Agent or Identified Group associated with this Profile document.	Required
since	Timestamp	Only ids of Profiles stored since the specified Timestamp (exclusive) are returned.	Optional

5.1.6.6 Activity Profile Resource (/activities/profile)

A place to store information about an Activity in a generic form called a document. This information is not tied to an Actor or registration. The semantics of the LRS response are driven by the presence of a "profileId" parameter. If it is included, the GET and method *shall* act upon a single defined profile document identified by "ProfileId". Otherwise, GET returns the available ids given through the other parameter. This resource has concurrency controls associated with it.

Example resource location/endpoint: http://example.com/xAPI/activities/profile

5.1.6.6.1 PUT Request

Summary: Stores a single document with the given id.

Body: The document to be stored.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

— An LRP making a PUT request to this resource *shall* include the "If-Match" header or the If- None-Match header.

5.1.6.6.2 POST Request

Summary: Stores/updates a single document with the given id.

Body: The document to be stored/updated.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

— If a Learning Record Provider needs to delete a property, it *should* use a PUT request to replace the whole document.

— An LRP making a POST request to this resource *shall* include the "If-Match" header or the If- None-Match header.

5.1.6.6.3 DELETE Request

Summary: Deletes a single document with the given id.

Body: None.

Returns: 204 No Content

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

An LRP making a DELETE request to this resource *shall* include the "If-Match" header.

5.1.6.6.4 (Single Document) GET Request

Summary: Fetches a single document with the given id.

Body: None.

Returns: 200 OK, the Profile document

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with this Profile document.	Required
profileId	String	The profile id associated with this Profile document.	Required

5.1.6.6.5 (Multiple Document) GET Request

Summary: Fetches Profile ids of all Profile documents for an Activity. If "since" parameter is specified, this is limited to entries that have been stored or updated since the specified Timestamp (exclusive).

Body: None.

Returns: 200 OK, Array of profileIds.

Parameter	Type	Description	Required
activityId	Activity id (IRI)	The Activity id associated with these Profile documents.	Required
since	Timestamp	Only ids of Profile documents stored since the specified Timestamp (exclusive) are returned.	Optional

5.1.6.7 About Resource (/about)

Returns JSON Object containing information about this LRS, including supported extensions and xAPI version(s).

Example resource location/endpoint: http://example.com/xAPI/about

5.1.6.7.1 GET Request

Body: None.

Returns: 200 OK, JSON object containing basic metadata about this LRS

Property	Type	Description	Required
version	Array of version strings	xAPI versions this LRS supports	Required
extensions	Object	A map of other properties as needed or supported additional resources (extensions)	Optional

5.1.7 Versioning

Using Semantic Versioning allows LRPs to remain interoperable as the specification changes.

Every request to the LRS and every response from the LRS *shall* include an HTTP header with the name X-Experience-API-Version and the version as the value. For example, X-Experience-API-Version: 2.0.0 for version 2.0.0;

LRP Requirements

— The LRP *shall* include the "X-Experience-API-Version" header in every request.

— The LRP *shall* set this header to the latest patch version.

5.1.8 Authentication

In order to support the varying security requirements of different environments, a specific authentication mechanism is not defined.

5.1.9 Security

Security is beyond the scope of this document and left to the individual LRS provider as an implementation detail. Implementors are encouraged to follow industry best practices, e.g., The HTTPS-Only Standard from the office of the White House CIO.

It is possible that security concerns may arise in the implementation of this specification, and implementers might choose to break a conformance requirement for the sake of security. In these cases, implementers are encouraged to consider both the security and interoperability implications of their implementation decisions.

5.2 LRP Data Requirements

The LRP is responsible for creation of various data formats described in this document. The most common data format in xAPI is the Statement.

An LRP *shall not* use a property in a Statement more than one time

The required field in each of these tables dictates the LRP responsibility in creating data. Required indicates the LRP *shall* include this property in the format supplied. Recommended indicates that the LRP *should* create the property, many of which are set by the LRS if not provided. Optional indicates that the field may be used by an LRP, but the LRS takes no default responsibility if it is not provided. Not Recommended indicates that the LRP *should not* use this property and instead the LRS should populate the value.

Examples of the data requirements can be found at https://opensource.ieee.org/xapi/xapi-base-standard-examples.

Statement Immutability:

Statements are inevitably stored in databases. While the methods to PUT/POST and GET them are specific, the storage mechanism is not. JSON has flexibility in form, which means that a reconstruction may not be exact, ordering being a very common example. This document defines only certain aspects of Statements to be immutable. For all intents and purposes, if immutability required fields are equal, then the Statements are equal.

- The following properties are immutable
  - Actor (except the ordering of group members)
  - Verb (except for display)
  - Object
  - Duration (see 5.2.7.6 for further requirements)
- The following properties are not immutable
  - Case
  - Id
  - Order of any Group Members
  - Authority
  - Stored
  - Timestamp
  - Version
  - Any attachments
  - Any referenced Activity Definitions

5.2.1 Table Guidelines

Tables in this document represent requirements that shall be followed. It is the responsibility of an LRP to create Statements and requests that contain data that follow requirements in these tables. The following requirements are expected to be followed for each table in this subclause.

NOTE—The "description" portion of table contains requirements, in particular for controlled vocabulary, enumerations, etc.

An LRP *shall not* add additional properties to Statements

An LRP *shall not* use a property more than once

The LRP *shall* create Statements:

- With all required properties
- Without any null values (except inside extensions).
- Following all formatting of the data type provided
- Following format of key or value, including not providing an empty string, where a string with a particular format (such as mailto IRI, UUID, or IRI) is required.

— Where the case of a key does match the case specified in this specification.

— Where the case of a value restricted to enumerated values matches an enumerated value given in this specification exactly.

— Which do not have a key or value that is not allowed by this specification.

— That contain IRL or IRI values with a scheme.

5.2.2 Statement

Each Statement in xAPI has the following JSON structure and requirements. Note that properties with type "Object" are detailed in subsequent tables.

Property	Type	Description	Required
id	UUID	UUID assigned by LRS if not set by the Learning Record Provider.	Recommended
actor	Object	Whom the Statement is about, as an Agent or Group Object.	Required
verb	Object	Action taken by the Actor.	Required
object	Object	Activity, Agent, or another Statement that is the Object of the Statement.	Required
result	Object	Result Object, further details representing a measured outcome.	Optional
context	Object	Context that gives the Statement more meaning. Examples: a team the Actor is working with, altitude at which a scenario was attempted in a flight simulator.	Optional
timestamp	Timestamp	Timestamp of when the events described within this Statement occurred. Set by the LRS if not provided.	Optional
stored	Timestamp	Timestamp of when this Statement was recorded.	Set by LRS
authority	Object	Agent or Group who is asserting this Statement is true. Verified by the LRS based on authentication. Set by LRS if not provided or if a strong trust relationship between the Learning Record Provider and LRS has not been established.	Optional
version	Version	The Statement's associated xAPI version, formatted according to Semantic Versioning 1.0.0.	Not Recommended
attachments	Ordered array of Attachment Objects	Headers for Attachments to the Statement	Optional

5.2.2.1 Actor

Actors can take on the form of either Groups or Agents. Groups can be identified, and if they are not, are considered to be anonymous groups. Specific requirements are found in the tables below.

Actor as Agent Table: An actor property that implements an Agent has the following JSON structure and requirements:

Property	Type	Description	Required
objectType	String	Value is "Agent". This property is optional except when the Agent is used as a Statement's object.	Optional (except when it is to be used in a Statement Object)
name	String	Full name of the Agent.	Optional
mbox	mailto IRI	The required format is "mailto:email address". Only email addresses uniquely assigned to this Agent, but no others, *should* be used for this property and mbox_sha1sum.	Exactly One of mbox, mbox_sha1sum, openid, account is required

Table continues

Property	Type	Description	Required
mbox_sha1sum	String	The hex-encoded SHA1 hash of a mailto IRI (i.e., the value of an mbox property). An LRS *may* include Agents with a matching hash when a request is based on an mbox.	Exactly One of mbox, mbox_sha1sum, openid, account is required
openid	URI	An openID that uniquely identifies the Agent.	Exactly One of mbox, mbox_sha1sum, openid, account is required
account	Object	A user account on an existing system e.g., an LMS or intranet.	Exactly One of mbox, mbox_sha1sum, openid, account is required

Actor as Anonymous Group Table: An actor property that represents an Anonymous Group *shall* adhere to the requirements in the following table.

Property	Type	Description	Required
objectType	String	Group.	Required
name	String	Name of the Group.	Optional
member	Array of Agent Objects	The members of this Group. This is an unordered list.	Required

Actor as Identified Group Table: An actor property that implements an Identified Group *shall* adhere to the requirements in the following table.

Property	Type	Description	Required
objectType	String	Group.	Required
name	String	Name of the Group.	Optional
member	Array of Agent Objects	The members of this Group. This is an unordered list.	Optional
mbox	mailto IRI	The required format is “mailto:email address”. Only email addresses uniquely assigned assigned to this Agent, but no others, *should* be used for this property and mbox_sha1sum.	Exactly One of mbox, mbox_sha1sum, openid, account is required
mbox_sha1sum	String	The hex-encoded SHA1 hash of a mailto IRI (i.e., the value of an mbox property). An LRS *may* include Agents with a matching hash when a request is based on an mbox.	Exactly One of mbox, mbox_sha1sum, openid, account is required
openid	URI	An openID that uniquely identifies the Agent.	Exactly One of mbox, mbox_sha1sum, openid, account is required
account	Object	A user account on an existing system e.g., an LMS or intranet.	Exactly One of mbox, mbox_sha1sum, openid, account is required

Account Table: Actors (Agents and Groups, including when they are Objects) may use the Account Object (as demonstrated in previous tables). If used, the properties of that Account Object are:

Property	Type	Description	Required
homePage	IRL	The canonical home page for the system the account is on. This is based on FOAF's accountServiceHomePage.	Required
name	String	The unique id or name used to log in to this account. This is based on FOAF's accountName.	Required

5.2.2.2 Verb

Verbs appear in Statements as Objects consisting of an IRI and a set of display names corresponding to multiple languages or dialects which provide human-readable meanings of the Verb. The table below lists all properties of the Verb Object.

Property	Type	Description	Required
id	IRI	Corresponds to a Verb definition. Each Verb definition corresponds to the meaning of a Verb, not the word.	Required
display	Language Map	The human readable representation of the Verb in one or more languages. This does not have any impact on the meaning of the Statement, but serves to give a human-readable display of the meaning already determined by the chosen Verb.	Recommended

5.2.2.3 Object

Objects in xAPI vary widely in use and are classified in that use by the objectType property. Each of the following subclauses provide additional requirements that stem from the use of objectType. That is, if an Object is to be used as an Activity, its objectType shall be "Activity".

Object As Activity Table: The format for an Activity Object is described in the table below:

Property	Type	Description	Required
objectType	String	*shall* be Activity when present	Required
id	IRI	An identifier for a single unique Activity	Required
definition	Object	Metadata	Optional

Activity Definition (referenced in Object as Activity) Table: All properties are optional, but if implemented include requirements described below.

Property	Type	Description	Required
name	Language Map	The human readable/visual name of the Activity	Recommended
description	Language Map	A description of the Activity	Recommended
type	IRI	The type of Activity.	Recommended
moreInfo	IRL	Resolves to a document with human-readable information about the Activity, which could include a way to launch the activity.	Optional
Interaction Activities	Object	Interaction definitions	Optional
extensions	Object	A map of other properties as needed	Optional

Interaction Activities (referenced in Activity Definition) Table: The format for Interaction Activities is described in the table below:

Property	Type	Description	Required
interactionType	String	The type of interaction. Possible values are: true-false, choice, fill-in, long-fill-in, matching, performance, sequencing, likert, numeric or other.	Required
correctResponsesPattern	Array of Strings	A pattern representing the correct response to the interaction. The structure of this pattern varies depending on the interactionType. This is detailed below.	Optional
choices, scale, source, target, or steps	Array of interaction components	Specific to the given interactionType (see below).	Optional

Interaction Types (referenced in Interaction Activities Table): The interactionType is controlled vocabulary. Based on the term used, the data formatting and purpose changes according to the following table.

interactionType	Description	Format
true-false	An interaction with two possible responses: true or false.	Either true or false
choice	An interaction with a number of possible choices from which the learner can select. This includes interactions in which the learner can select only one answer from the list and those where the learner can select multiple items.	A list of item ids delimited by [,]. If the response contains only one item, the delimiter *shall* not be used.
fill-in	An interaction which requires the learner to supply a short response in the form of one or more strings of characters.Typically, the correct response consists of part of a word, one word or a few words. "Short" means that the correct responses pattern and learner response strings are normally 250 characters or less.	A list of responses delimited by [,]. If the response contains only one item, the delimiter *shall* not be used.
long-fill-in	An interaction which requires the learner to supply a response in the form of a long string of characters. "Long" means that the correct responses pattern and learner response strings are normally more than 250 characters.	A list of responses delimited by [,]. If the response contains only one item, the delimiter *shall* not be used.
matching	An interaction where the learner is asked to match items in one set (the source set) to items in another set (the target set). Items do not have to pair off exactly and it is possible for multiple or zero source items to be matched to a given target and vice versa.	A list of matching pairs, where each pair consists of a source item id followed by a target item id. Items can appear in multiple (or zero) pairs. Items within a pair are delimited by [.]. Pairs are delimited by [,].
performance	An interaction that requires the learner to perform a task that requires multiple steps.	A list of steps containing a step ids and the response to that step. Step ids are separated from responses by [.]. Steps are delimited by [,]. The response can be a String as in a fill-in interaction or a number range as in a numeric interaction.
sequencing	An interaction where the learner is asked to order items in a set.	An ordered list of item ids delimited by [,].

Table continues

interactionType	Description	Format
likert	An interaction which asks the learner to select from a discrete set of choices on a scale	A single item id
numeric	Any interaction which requires a numeric response from the learner.	A range of numbers represented by a minimum and a maximum delimited by [:]. Where the range does not have a maximum or does not have a minimum, that number is omitted but the delimiter is still used. E.g. [:]4 indicates a maximum for 4 and no minimum. Where the correct response or learner's response is a single number rather than a range, the single number with no delimiter may be used.
other	Another type of interaction that does not fit into those defined above.	Any format is valid within this string as appropriate for the type of interaction.

The Correct Responses Pattern contains an array of response patterns. A learner's response is be considered correct if it matches *any* of the response patterns in that array. Where a response pattern is a delimited list, the learner's response is only considered correct if *all* of the items in that list match the learner's response.

Characterstring parameters

Some of the values within the responses described above can be prepended with certain additional parameters. These parameters are represented by the format {parameter = value}.

Characterstring parameters are not validated by the LRS. Systems interpreting Statement data can use their best judgement in interpreting (or ignoring) invalid characterstring parameters and values.

The following parameters are valid at the start of the string representing the list of items for the listed interaction types:

Parameter	Default	Description	Value	Interaction types
case_matters	false	Whether or not the case of items in the list matters.	true or false	fill-in, long-fill-in
order_matters	true	Whether or not the order of items in the list matters.	true or false	fill-in, long-fill-in, performance

The following parameters are valid at the start of each item in the list for the listed interaction types:

Parameter	Description	Value	Interaction types
lang	The language used within the item.	RFC 5646 Language Tag	fill-in, long-fill-in, performance (String responses only)

Interaction Components, Expanded (part of the Interaction Activities Table): Depending on interactionType, Interaction Activities can take additional properties, each containing a list of interaction components. These additional properties are called "interaction component lists". The following table shows the supported interaction component list(s) for an Interaction Activity with the given interactionType.

interactionType	Interaction Component Lists	Description
choice, sequencing	choices	A list of the options available in the interaction for selection or ordering.
likert	scale	A list of the options on the likert scale.

Table continues

interactionType	Interaction Component Lists	Description
matching	source, target	Lists of sources and targets to be matched.
performance	steps	A list of the elements making up the performance interaction.
true-false, fill-in, long-fill- in, numeric, other	[No component lists supported]	None

Regardless of interactionType, each Component List has the following additional properties:

Property	Type	Description	Required
id	String	Identifies the interaction component within the list.	Required
description	Language Map	A description of the interaction component (for example, the text for a given choice in a multiple- choice interaction)	Optional

Object As Actor Table: The table below lists all properties of an Actor Object

Property	Type	Description	Required
objectType	String	*shall* be Agent or Group	Required
See 4.2.2.1 for addtional details regarding Agent properties.			

Object As Statement Table: The table below lists all properties of a Statement Reference Object

Property	Type	Description	Required
objectType	String	In this case, *shall* be StatementRef.	Required
id	UUID	The UUID of a Statement.	Required

Object As Sub-Statement Table: The table below lists all properties of a Sub-Statement Object

Property	Type	Description	Required
objectType	String	In this case, *shall* be SubStatement.	Required
actor	Object	Whom the Statement is about, as an Agent or Group Object.	Required
verb	Object	Action taken by the Actor.	Required
object	Object	Activity, Agent, or another Statement that is the Object of the Statement.	Required
result	Object	Result Object, further details representing a measured outcome.	Optional
context	Object	Context that gives the Statement more meaning. Examples: a team the Actor is working with, altitude at which a scenario was attempted in a flight simulator.	Optional
timestamp	Timestamp	Timestamp of when the events described within this Statement occurred. Set by the LRS if not provided.	Optional
attachments	Ordered array of Attachment Objects	Headers for Attachments to the Statement	Optional

5.2.2.4 Result

Represents a measured outcome related to the Statement in which it is included.

Result Table: The table below lists all properties of the Result Object.

Property	Type	Description	Required
score	Object	The score of the Agent in relation to the success or quality of the experience.	Optional
success	Boolean	Indicates whether or not the attempt on the Activity was successful.	Optional
completion	Boolean	Indicates whether or not the Activity was completed.	Optional
response	String	A response appropriately formatted for the given Activity.	Optional
duration	Duration	Period of time over which the Statement occurred.	Optional
extensions	Object	A map of other properties as needed.	Optional

Score Table: Represents the outcome of a graded Activity achieved by an Agent. The table below lists all properties of the Score Object.

Property	Type	Description	Required
scaled	Decimal number between −1 and 1, inclusive	The score related to the experience as modified by scaling and/or normalization.	Recommended
raw	Decimal number between min and max (if present, otherwise unrestricted), inclusive	The score achieved by the Actor in the experience described by the Statement. This is not modified by any scaling or normalization.	Optional
min	Decimal number less than max (if present)	The lowest possible score for the experience described by the Statement.	Optional
max	Decimal number greater than min (if present)	The highest possible score for the experience described by the Statement.	Optional

5.2.2.5 Context

Property to add contextual information to a Statement. It can store information such as the instructor for an experience, if this experience happened as part of a team-based Activity, or how an experience fits into some broader activity.

Context Table: The table below lists all properties of the context Object.

Property	Type	Description	Required
registration	UUID	The registration that the Statement is associated with.	Optional
instructor	Agent (*may* be a Group)	Instructor that the Statement relates to, if not included as the Actor of the Statement.	Not Recommended
team	Group	Team that this Statement relates to, if not included as the Actor of the Statement.	Not Recommended
contextActivities	contextActivities Object	A map of the types of learning activity context that this Statement is related to. Every key in the contextActivities Object *shall* be one of "parent", "grouping", "category", or "other". Every value in the contextActivities Object *shall* be either a single Activity Object or an array of Activity Objects.	Optional
contextAgents	Array of contextAgent Objects	Collection of Objects describing relationship(s) between Agent(s) and the current Statement. Zero or more Relevant Type IRIs are used to categorize these relationship(s).	Optional

Table continues

Property	Type	Description	Required
contextGroups	Array of contextGroup Objects	Collection of Objects describing relationship(s) between Identified or Anonymous Group(s) and the current Statement. Zero or more Relevant Type IRIs are used to categorize these relationship(s).	Optional
revision	String	Revision of the learning activity associated with this Statement. Format is free.	Optional
platform	String	Platform used in the experience of this learning activity.	Optional
language	String (as defined in RFC 5646)	Code representing the language in which the experience being recorded in this Statement (mainly) occurred in, if applicable and known.	Optional
statement	Statement Reference	Another Statement to be considered as context for this Statement.	Optional
extensions	Object	A map of any other domain-specific context relevant to this Statement. For example, in a flight simulator altitude, airspeed, wind, attitude, GPS coordinates might all be relevant.	Optional

NOTE—With the addition of context agents and context groups, it is recommended not to use instructor or team. They are supported in this document for backward compatibility purposes.

Context Agents and Groups Details:

A single Statement may require the inclusion of many contextually relevant Agent(s) and/or Group(s) in order to properly describe an experience. When this is the case, the relationship between the Agent(s) and/or Group(s) and the Statement itself needs to be represented in a structured manner.

The context properties contextAgents and contextGroups serve as this structure.

- Inclusion of contextAgents within a Statement establishes that a relationship exists between said Statement and one or more Agent(s)
- Inclusion of contextGroups within a Statement establishes that a relationship exists between said Statement and one or more Group(s)
- Zero or more Relevant Types are used to categorize each Statement specific relationship
- Each Statement specific relationship corresponds to an individual contextAgents or contextGroups Object

The relationship established by each contextAgents and contextGroups Object is limited in scope to the Statement in which the Object is found. In general, an Agent many have permanent characteristics, characteristics which are consistent across experiences, but these kinds of Agent specific properties should be captured in an Agent Profile

All Objects found within the contextAgents and/or contextGroups array(s) are independent of one another.

Context Agents Table:

Each contextAgent Object found within a contextAgents array has the following properties:

Property	Type	Description	Required
objectType	String	contextAgent	Required
agent	Agent Object	A single Agent Object for which a Statement specific relationship is being defined	Required
relevantTypes	Array of Relevant "type" IRIs	A collection of 1 or more Relevant Type(s) used to characterize the relationship between the Statement and the Agent. If not provided, only a generic relationship is intended (not recommended)	Optional

— Any and All valid Agent Objects can be used as the agent within a contextAgent Object.

— An Agent Object does *not* need to be found within any other Statement property in order to be included as the agent property within a contextAgent Object.

— Any and All valid Relevant Type IRIs can be included within the relevantTypes array of a contextAgent Object.

— A Relevant Type IRI does *not* need to be found within any other Statement property in order to be included within the relevantTypes property of a contextAgent Object

Context Group Table:

Each contextGroup Object found within a contextGroups array has the following properties:

Property	Type	Description	Required
objectType	String	contextGroup	Required
group	Group	A single Group Object for which a Statement specific relationship is being defined.	Required
relevantTypes	Array of Relevant "type" IRIs	A collection of 1 or more Relevant Type(s) used to characterize the relationship between the Statement and the Agent. If not provided, only a generic relationship is intended (not recommended).	Optional

— Any and All valid Group Objects can be used as the group within a contextGroup Object

— A Group Object does *not* need to be found within any other Statement property in order to be included as the group property within a contextGroup Object.

— Any and All valid Relevant Type IRIs can be included within the relevantTypes array of a contextAgent Object.

— An Relevant Type IRI does *not* need to be found within any other Statement property in order to be included within the relevantTypes property of a contextAgent Object

5.2.2.6 Attachments

In some cases an Attachment is logically an important part of a Learning Record. It could be an essay, a video, etc. Another example of such an Attachment is (the image of) a certificate that was granted as a result of an experience.

Attachments Table: The table below lists all properties of the attachments Object.

Property	Type	Description	Required	Corresponding Request Parameter
usageType	IRI	Identifies the usage of this Attachment. For example: one expected use case for Attachments is to include a "completion certificate". An IRI corresponding to this usage *shall* be coined, and used with completion certificate attachments.	Required	
display	Language Map	Display name (title) of this Attachment.	Required	
description	Language Map	A description of the Attachment.	Optional	
contentType	Internet Media Type	The content type of the Attachment.	Required	Content-Type
length	Integer	The length of the Attachment data in octets.	Required	Content-Length
sha2	String	The SHA-2 hash of the Attachment data. This property is always required, even if fileUrl is also specified.	Required	X-Experience-API-Hash
fileUrl	IRL	An IRL at which the Attachment data can be retrieved, or from which it used to be retrievable.	Optional	

5.2.3 Metadata

Metadata is additional information about a resource. It enables decision making, search, and discoverability. In xAPI, metadata can be utilized essentially in two ways.

First, metadata is used in the definition property of an Activity Object (the Object when the objectType is Activity). In this case, the metadata is part of the Statement. Other fields, including extensions, can be used as metadata in a similar regard. Including metadata in a Statement allows metadata about the IRI to be expressed without the necessity of resolving it.

Second, metadata may be hosted. In this case, additional information about an identifier can be provided within a Statement and can be hosted at the location pointed to by the identifier IRI.

Hosting metadata at the IRI location allows the owner of the IRI to define the canonical metadata for that IRI. This is the main purpose for which IRIs are used in xAPI.

All hosted metadata, including the format, is optional. However, it is recommended that the Activity Definition Object metadata is followed for hosted Activity identifiers.

For the structure of metadata about all other identifiers, see the format below:

Property	Type	Description	Required
name	Language Map	The human readable/visual name. For Verbs, this is equivalent to the "display" property in a Statement.	Optional
description	Language Map	Description	Optional

Hosted metadata consists of a document containing a JSON object as described above. If this hosted metadata is provided, it is the canonical source of information about the identifier it describes.

5.2.3.1 Learning Record Provider Recommendations

The following recommendations are made for any LRP that wishes to implement resolvable metadata for querying purposes.

- Metadata *may* be provided with an identifier.
- If metadata is provided, both "name" and "description" *should* be included.
- For any of the identifier IRIs above the Metadata Provider *should* make a human-readable description of the intended usage accessible at the IRI.
- For any of the identifier IRIs above the Metadata Provider *should* ensure that this JSON metadata available at that IRI when the IRI is requested and a Content-Type of application/json is requested.
- Where the IRI represents an Activity, the Metadata Provider *may* host metadata using the Activity Definition JSON format which is used in Statements, with a Content-Type of application/json.

5.2.4 LRP Processing of Data

An LRP functions as the creator of xAPI Data. Statements, in particular, are highly structured and have many requirements. An LRP is responsible for formatting and sending statements such that they do not get rejected by the LRS, and otherwise follow Data Design Best Practices

Statements are expected to be unique and to be immutable. In this regard they are unchangeable and should not be deleted. This subclause outlines requirements that focus on specific practices that are unique to certain Statement properties.

5.2.4.1 LRP Statement Best Practices

Some of the requirements in this subclause are covered in the tables above but are explained in more detail below or are listed again to reiterate best practices.

- An LRP *should* provide an id for the Statement
- An LRP *should not* provide a stored for the Statement
- An LRP *should not* provide authority for the Statement
- An LRP *should not* provide a version for the Statement
- If an LRP sets the Statement version, it *shall* set it to 2.0.0
- An LRP *should not* include properties with a value of an empty object.
- The Learning Record Provider *should* ensure that every value in the contextActivities Object is an array of Activity Objects instead of a single Activity Object.
- A Learning Record Provider *shall not* use a future time for a "timestamp" property in a Statement.
- An LRP *may* create a Statement where a SubStatement has a "timestamp" property that is in the future.
- The LRP *shall not* create a Statement with an Object with ObjectType SubStatement that has within it an Object with ObjectType SubStatement.
- An LRP *should not* provide an interaction component's id value with whitespace in the value.
- The LRP *shall not* provide a "language" property with invalid values for any reason, including if not applicable or unknown.

5.2.5 Statement Voiding

Not all Statements are perpetually valid once they have been issued. Mistakes or other factors could dictate that a previously made Statement is marked as invalid. This is called "voiding a Statement" and the reserved Verb http://adlnet.gov/expapi/verbs/voided is used for this purpose. Any Statement that voids another cannot itself be voided.

An LRP *shall not* send a Statement with the Verb http://adlnet.gov/expapi/verbs/voided when:

— The Statement's Object's objectType property is not "StatementRef"

— The Statement has no "id" for the Object (e.g., no target for voiding)

5.2.6 Statement Signing

A Statement can include a digital signature to provide strong and durable evidence of the authenticity and integrity of the Statement.

Signed Statements include a JSON web signature (JWS) as an Attachment. This allows the original serialization of the Statement to be included along with the signature. For interoperability, the "RSA + SHA" series of JWS algorithms have been selected, and for discoverability of the signer X.509 certificates *should* be used.

Statement Signing Process:

— A Signed Statement *shall* include a JSON web signature (JWS) as defined here: http://tools.ietf.org/html/rfc7515, as an Attachment with a usageType of http://adlnet.gov/expapi/attachments/signatureandacontentTypeofapplication/octet-stream.

— JWS Compact Serialization *shall* be used to create the JSON web signature. Use of JWS JSON Serialization is strongly discouraged.

— The JWS signature *shall* have a payload of a valid JSON serialization of the complete Statement before the signature was added.

— The JWS signature *shall* use an algorithm of "RS256", "RS384", or "RS512".

— The JWS signature *should* have been created based on the private key associated with an X.509 certificate.

— If X.509 was used to sign, the JWS header *should* include the "x5c" property containing the associated certificate chain.

5.2.7 Additional Requirements for Data Types

The following subclauses provide guidance and requirements for data types found in this document. Many tables contain specific data types that have requirements found in this subclause.

5.2.7.1 IRIs

Internationalized Resource Identifiers, or IRIs, are unique identifiers which could also be resolvable. Because resolving is not a requirement, IRIs/URIs are used instead of IRLs/URLs. In order to allow the greatest flexibility in the characters used in an identifier, IRIs are used instead of URIs as IRIs can contain some characters outside of the ASCII character set.

The LRP has responsibilities in regard to each IRI as outlined below. Many of these are only best practices as they are not testable through software, but if not followed, undermine the use of xAPI in a larger environment. To put it another way, the *should* requirements carry the weight of a *shall* requirement.

— Learning Record Providers defining new IRIs *should* only use IRIs they control or have permission from the controller to use.

— Where a suitable identifier already exists, the LRP *should* use the corresponding existing identifier and *should not* create a new identifier.

— When re-using an existing identifier, Learning Record Providers *should* ensure that the exact character equivalent IRI is used.

— A Learning Record Provider *may* create their own identifiers where a suitable identifier does not already exist.

— When defining identifiers, the Learning Record Provider *may* use IRIs containing anchors so that a single page can contain definitions for multiple identifiers. E.g. http://example.com/xapi/verbs#defenestrated.

— When defining identifiers, the LRP *should* use lowercase IRIs.

5.2.7.2 Extensions

Extensions are available as part of Activity Definitions, as part of a Statement's "context" property, or as part of a Statement's "result" property. In each case, extensions are intended to provide a natural way to extend those properties for some specialized use. The contents of these extensions might be something valuable to just one application, or it might be a convention used by an entire Community of Practice.

Extensions are defined by a map and logically relate to the part of the Statement where they are present. The values of an extension can be any JSON value or data structure. Extensions in the "context" property provide context to the core experience, while those in the "result" property provide elements related to some outcome. Within Activities, extensions provide additional information that helps define an Activity within some custom application or Community of Practice. The meaning and structure of extension values under an IRI key are defined by the person who controls the IRI.

— The keys of an extensions map *shall* be IRIs.

— Learning Record Providers *should* always strive to map as much information as possible into the built-in elements in order to leverage interoperability among xAPI conformant tools.

— All extension IRIs *should* have controllers.

— The controller of an IRL extension key *should* make a human-readable description of the intended meaning of the extension supported by the IRL accessible at the IRL.

5.2.7.3 Language Maps

A language map is a dictionary where the key is an RFC 5646 Language Tag, and the value is a string in the language specified in the tag. This map *should* be populated as fully as possible based on the knowledge of the string in question in different languages.

The shortest valid language code for each language string is generally preferred. The ISO 639 language code plus an ISO 3166-1 country code allows for the designation of basic languages (e.g., es for Spanish) and regions (e.g., es-MX, the dialect of Spanish spoken in Mexico). If only the ISO 639 language code is known for certain, do not guess at the possible ISO 3166-1 country code. For example, if only the primary language is known (e.g., English) then use the top-level language tag en, rather than en-US. If the specific regional variation is known, then use the full language code.

NOTE—For Chinese languages, the significant linguistic diversity represented by zh means that the ISO 639 language code is generally insufficient.

The content of strings within a language map is plain text. It is expected that any formatting code such as HTML tags or markdown is not rendered but displayed as code when this string is displayed to an end user. An important exception to this is if language map Object is used in an extension and the owner of that extension IRI explicitly states that a particular form of code is to be rendered.

An LRP *shall* only send a Statement with invalid RFC 5646 language tags (Keys of language maps *shall* be sent with valid RFC 5646 language tags, for similar reasons, The LRS *shall* at least validate that the sequence of token lengths for language map keys matches the RFC 5646 standard)

5.2.7.4 UUIDs

Universally Unique Identifiers, or UUIDs, are 128-bit values that are globally unique. Unlike IRIs, there is no expectation of resolvability as UUIDs take on a completely different format. UUIDs *shall* be in the standard string form. It is recommended variant 2 in RFC 4122 is used.

5.2.7.5 Timestamps

— A Timestamp *shall* be formatted according to ISO 8601.

— A Timestamp *shall* be expressed using the format described in RFC 3339, which is a profile of ISO 8601.

— A Timestamp *shall* be formatted to UTC.

— If used, An LRP *should* send a Timestamp with at least millisecond precision (3 decimal points beyond seconds).

5.2.7.6 Duration

— An LRP *shall* provide Statements with Duration expressed using the format for Duration in ISO 8601:2004(E) section 4.4.3.2.

— An LRP *shall not* provide Statements with the alternative format (in conformity with the format used for time points and described in ISO 8601:2004(E) section 4.4.3.3).

— Learning Record Providers *should* provide a maximum precision of 0.01 s.

— Learning Record Providers *may* provide less precision, for example in the case of reading a University Degree precision might be in months or years.

— When comparing Durations (or Statements containing them), any precision beyond 0.01 s precision *shall not* be included in the comparison.

5.2.7.7 Documents

Note that the following table shows generic properties, not a JSON Object as many other tables in this specification do. The id is stored in the IRL, “updated” is HTTP header information, and “contents” is the HTTP document itself (as opposed to an Object).

Property	Type	Description
id	String	Set by Learning Record Provider, unique within the scope of the Agent or Activity.
updated	Timestamp	When the document was most recently modified (HTTP Header).
contents	Arbitrary binary data	The contents of the document.

Annex A

(informative)

xAPI Base Standard Examples

A.1 Example Statements

Example of a simple Statement (line breaks are for display purposes only):

```
{
       "id":"fd41c918-b88b-4b20-a0a5-a4c32391aaa0"
       "timestamp": "2015-11-18T12:17:00+00:00",
       "actor":{
              "objectType": "Agent",
              "name":"Project Tin Can API",
              "mbox":"mailto:user@example.com"
       },
       "verb":{
              "id":"http://example.com/xapi/verbs#sent-a-statement",
              "display":{
                     "en-US":"sent"
              }
       },
       "object":{

"id":"http://example.com/xapi/activity/simplestatement",
              "definition":{
                     "name":{
                            "en-US":"simple statement"
                     },
                     "description":{
                            "en-US":"A simple Experience API statement.
Note that the LRS does not need to have any prior information about the Actor
(learner), the verb, or the Activity/object."
                     }
              }
       }
}
```

Completion with Verb named “attempted” and Duration expressed in seconds (not converted to minutes and seconds):

```
{
       "id":"7ccd3322-e1a5-411a-a67d-6a735c76f119",
       "timestamp": "2015-12-18T12:17:00+00:00",
       "actor":{
              "objectType": "Agent",
              "name":"Example Learner",
              "mbox":"mailto:example.learner@adlnet.gov"
       },
       "verb":{
              "id":"http://adlnet.gov/expapi/verbs/attempted",
              "display":{
        "en-US":"attempted"
```

```
        }
 },
 "object":{
"id":"http://example.adlnet.gov/xapi/example/simpleCBT",
        "definition":{
                "name":{
                        "en-US":"simple CBT course"
        },
        "description":{
                "en-US":"A fictitious example CBT course."
        }
 }
},
"result":{
        "score":{
                "scaled":0.95
        },
        "success":true,
        "completion":true,
        "duration": "PT1234S"
        }
}
```

A long example Statement showcasing most of the properties available. This example shows a Statement returned by an LRS including the “authority” and “stored” properties set by the LRS:

```
{
        "id": "6690e6c9-3ef0-4ed3-8b37-7f3964730bee",
        "actor": {
                "name": "Team PB",
                "mbox": "mailto:teampb@example.com",
                "member": [
        {
                "name": "Andrew Downes",
                "account": {
                        "homePage": "http://www.example.com",
                        "name": "13936749"
                },
                "objectType": "Agent"
        },
        {
                "name": "Toby Nichols",
                "openid": "http://toby.openid.example.org/",
                "objectType": "Agent"
        },
        {
                "name": "Ena Hills",
                "mbox_sha1sum":
"ebd31e95054c018b10727ccffd2ef2ec3a016ee9",
                "objectType": "Agent"
        }
                ],
                "objectType": "Group"
        },
        "verb": {
```

```
              "id": "http://adlnet.gov/expapi/verbs/attended",
              "display": {
              "en-GB": "attended",
              "en-US": "attended"
          }
        },
        "result": {
          "extensions": {
"http://example.com/profiles/meetings/resultextensions/minuteslocation": "X:\\
meetings\\minutes\\examplemeeting.one"
        },
          "success": true,
          "completion": true,
          "response": "We agreed on some example actions.",
          "duration": "PT1H0M0S"
 },
 "context": {
        "registration": "ec531277-b57b-4c15-8d91-d292c5b2b8f7",
        "contextActivities": {
          "parent": [
           {
              "id":
 "http://www.example.com/meetings/series/267",
              "objectType": "Activity"
           }
         ],
         "category": [
           {
              "id":
"http://www.example.com/meetings/categories/teammeeting",
              "objectType": "Activity",
              "definition": {
                 "name": {
                     "en": "team meeting"
                 },
                 "description": {
                     "en": "A category of meeting
used for regular team meetings."
              },
              "type":
"http://example.com/expapi/activities/meetingcategory"
               }
              }
        ],
        "other": [
              {
                 "id":
"http://www.example.com/meetings/occurances/34257",
                 "objectType": "Activity"
              },
              {
                 "id":
"http://www.example.com/meetings/occurances/3425567",
                 "objectType": "Activity"
```

```
            }
          ]
        },
        "instructor" :
        {
          "name": "Andrew Downes",
          "account": {
               "homePage": "http://www.example.com",
               "name": "13936749"
          },
          "objectType": "Agent"
        },
        "team":
        {
               "name": "Team PB",
               "mbox": "mailto:teampb@example.com",
               "objectType": "Group"
        },
        "platform" : "Example virtual meeting software",
        "language" : "tlh",
        "statement" : {
           "objectType":"StatementRef",
           "id" :"6690e6c9-3ef0-4ed3-8b37-7f3964730bee"
          }
        },
        "timestamp": "2013-05-18T05:32:34.804+00:00",
        "stored": "2013-05-18T05:32:34.804+00:00",
        "authority": {
          "account": {
            "homePage": "http://cloud.scorm.com/",
            "name": "anonymous"
         },
         "objectType": "Agent"
        },
        "version": "1.0.0",
        "object": {
           "id":
"http://www.example.com/meetings/occurances/34534",
       "definition": {
           "extensions": {
"http://example.com/profiles/meetings/activitydefinitionextensions/room":
{"name": "Kilby", "id" :
"http://example.com/rooms/342"}
          },
          "name": {
              "en-GB": "example meeting",
              "en-US": "example meeting"
       },
       "description": {
          "en-GB": "An example meeting that happened on a specific  occasion
with certain people present.",
          "en-US": "An example meeting that happened on a specific occasion with
certain people present."
       },
```

```
       "type":
"http://adlnet.gov/expapi/activities/meeting",
       "moreInfo":
"http://virtualmeeting.example.com/345256"
       },
       "objectType": "Activity"
   }
}
```

A.2 Examples of Statement's Objects of different types

The Object of a Statement can be an Activity, an Agent, a Group or a Statement. This appendix provides one example of each.

A.2.1 Object is Activity

```
 {
       "id": "http://www.example.co.uk/exampleactivity",
       "definition": {
          "name": {
              "en-GB": "example activity",
              "en-US": "example activity"
       },
       "description": {
              "en-GB": "An example of an activity",
              "en-US": "An example of an activity"
       },
       "type":
"http://www.example.co.uk/types/exampleactivitytype"
  },
  "objectType": "Activity"
}
```

A.2.2 Object is Agent

```
{
       "name": "Andrew Downes",
       "mbox": andrew@example.co.uk
       "objectType": "Agent"
}
```

A.2.3 Object is Group

This example shows an Identified Group with members.

```
{
"name": "Example Group",
"account" : {
       "homePage" : "http://example.com/homePage",
       "name" : "GroupAccount"
},
"objectType": "Group",
"member": [
```

```
        {
                "name": "Andrew Downes",
                "mbox": "mailto:andrew@example.com",
                "objectType": "Agent"
        },
        {
                "name": "Aaron Silvers",
                "openid": "http://aaron.openid.example.org",
                "objectType": "Agent"
        }
    ]
}
```

A.2.4 Object is Statement

This example shows a SubStatement Object whose Object is a Statement Reference.

```
{
        "objectType": "SubStatement",
        "actor" : {
        "objectType": "Agent",
        "mbox":"mailto:agent@example.com"
},
"verb" : {
        "id":"http://example.com/confirmed",
        "display":{
                "en":"confirmed"
        }
 },
"object": {
        "objectType":"StatementRef",
        "id" :"9e13cefd-53d3-4eac-b5ed-2cf6693903bb"
 }
}
```

A.3 Example definitions for Activities of type cmi.interaction

A.3.1 true-false

```
"definition": {
  "description": {
        "en-US": "Does the xAPI include the concept of
statements?"
        },
 "type":
 "http://adlnet.gov/expapi/activities/cmi.interaction",
        "interactionType": "true-false",
        "correctResponsesPattern": [
        "true"
 ]
}
```

A.3.2 choice

```
"definition": {
  "description": {
       "en-US": "Which of these prototypes are available at
the beta site?"
  },
  "type":
"http://adlnet.gov/expapi/activities/cmi.interaction",
  "interactionType": "choice",
"correctResponsesPattern": [
"golf[,]tetris"
  ],
  "choices": [
  {
"id": "golf",
"description": {
  "en-US": "Golf Example"
}
  },
{
       "id": "facebook",
       "description": {
          "en-US": "Facebook App"
       }
  },
  {
       "id": "tetris",
       "description": {
         "en-US": "Tetris Example"
       }
  },
{
       "id": "scrabble",
       "description": {
         "en-US": "Scrabble Example"
    }
   }
  ]
}
```

A.3.3 fill-in

```
"definition": {
   "description": {
      "en-US": "Ben is often heard saying: "
    },
    "type":
"http://adlnet.gov/expapi/activities/cmi.interaction",
 "interactionType": "fill-in",
  "correctResponsesPattern": [
       "Bob's your uncle"
    ]
}
```

A.3.4 long-fill-in

```
"definition": {
 "description": {
    "en-US": "What is the purpose of the xAPI?"
  },
   "type":
"http://adlnet.gov/expapi/activities/cmi.interaction",
    "interactionType": "long-fill-in",
    "correctResponsesPattern": [
   "{case_matters=false}{lang=en}To store and provide
access to learning experiences."
  ]
}
```

A.3.5 likert

```
"definition": {
   "description": {
       "en-US": "How awesome is Experience API?"
   },
   "type":
   "http://adlnet.gov/expapi/activities/cmi.interaction",
  "interactionType": "likert",
"correctResponsesPattern": [
  "likert_3"
],
"scale": [
   {
"id": "likert_0",
"description": {
   "en-US": "It's OK"
      }
  },
{
     "id": "likert_1",
     "description": {
         "en-US": "It's Pretty Cool"
    }
},
{
     "id": "likert_2",
     "description": {
        "en-US": "It's Damn Cool"
      }
},
{
      "id": "likert_3",
      "description": {
        "en-US": "It's Gonna Change the World"
      }
  }
 ]
}
```

A.3.6 matching

```
"definition": {
   "description": {
     "en-US": "Match these people to their kickball

team:"
},
"type": "http://adlnet.gov/expapi/activities/cmi.interaction",
       "interactionType": "matching", "correctResponsesPattern": [
       "ben[.]3[,]chris[.]2[,]troy[.]4[,]freddie[.]1"
 ],
       "source": [
{
       "id": "ben",
"description": {
"en-US": "Ben"
       }
      },
   {
       "id": "chris",
       "description": {
       "en-US": "Chris"
    }
},
{
"id": "troy",
       "description": {
         "en-US": "Troy"
       }
    },
{
"id": "freddie",
        "description": {
   "en-US": "Freddie"
}
     }
],
"target": [
    {
       "id": "1",
"description": {
   "en-US": "Swift Kick in the Grass"
  }
},
{
    "id": "2",
     "description": {
      "en-US": "We got Runs"
    }
},
{
     "id": "3",
     "description": {
```

```
        "en-US": "Duck"
      }
    },
{
"id": "4",
    "description": {
    "en-US": "Van Delay Industries"
  }
 }
]
}
```

A.3.7 performance

```
   "definition": {
   "description": {
       "en-US": "This interaction measures performance over
   a day of RS sports:"
},
   "type":
"http://adlnet.gov/expapi/activities/cmi.interaction",
    "interactionType": "performance",
    "correctResponsesPattern": [

     "pong[.]1:[,]dg[.]:10[,]lunch[.]"
],
    "steps": [
{
      "id": "pong",
      "description": {
           "en-US": "Net pong matches won"
    }
   },
{
"id": "dg",
    "description": {
     "en-US": "Strokes over par in disc golf at Liberty"
    }
  },
{
     "id": "lunch",
      "description": {
   "  en-US": "Lunch having been eaten"
     }
   }
 ]
}
```

A.3.8 sequencing

```
"definition": {
   "description": {
"en-US": "Order players by their pong ladder
```

```
 position:"
   },
   "type":
"http://adlnet.gov/expapi/activities/cmi.interaction",
    "interactionType": "sequencing",
    "correctResponsesPattern": [

    "tim[,]mike[,]ells[,]ben"
],
"choices": [
    {
      "id": "tim",
       "description": {
         "en-US": "Tim"
   }
  },
 {
      "id": "ben",
      "description": {
        "en-US": "Ben"
      }
   },
 {
   "id": "ells",
   "description": {
   "en-US": "Ells"
  }
  },
 {
     "id": "mike",
      "description": {
        "en-US": "Mike"
   }
  }
 ]
}
```

A.3.9 numeric

```
 "definition": {
     "description": {
        "en-US": "How many jokes is Chris the butt of each
day?"
    },
  "type":
     "http://adlnet.gov/expapi/activities/cmi.interaction",
        "interactionType": "numeric",
         "correctResponsesPattern": [
    "4[:]"
  ]
}
```

In this example the minimum correct answer is 4 and there is no maximum. 5, 6 or 976 would all be correct answers.

A.3.10 other

```
"definition": {
"description": {
"en-US": "On this map, please mark Franklin, TN"
},
"type":
"http://adlnet.gov/expapi/activities/cmi.interaction",
    "interactionType": "other",
    "correctResponsesPattern": [
  "(35.937432,-86.868896)"
  ]
}
```

A.4 Example Signed Statement

An example signed Statement, as described in: 4.2.6.

The original Statement serialization to be signed. New lines in this example are included via CR+LF (0x0D + 0x0A).

```
{
"version": "1.0.0",
"id": "33cff416-e331-4c9d-969e-5373a1756120",
"actor": {
     "mbox": "mailto:example@example.com",
     "name": "Example Learner",
     "objectType": "Agent"
},
"verb": {
    "id": "http://adlnet.gov/expapi/verbs/experienced",
    "display": {
       "en-US": "experienced"
    }
  },
"object": {
   "id": "https://www.youtube.com/watch?v=xh4kIiH3Sm8",
   "objectType": "Activity",
   "definition": {
"name": {
     "en-US": "Tax Tips & Information : How to
    File a Tax Return "
      },
       "description": {
   "en-US": "Filing a tax return will require
filling out either a 1040, 1040A or 1040EZ form"
      }
   }
},
"timestamp": "2013-04-01T12:00:00Z"
}
```

A.4.1 Example private key for X.509 certificate used for signing

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDjxvZXF30WL4oKjZYXgR0ZyaX+u3y6+JqTqiNkFa/VTnet
6Ly2
OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkW13j06j5iNn4obcCVWTL9yXNvJH+Ko+
xu4Y
l/ySPRrIPyTjtHdG0M2XzIlmmLqm+CAS+KCbJeH4tf543kIWC5pC5p3cVQID
AQAB
AoGAOejdvGq2XKuddu1kWXl0Aphn4YmdPpPyCNTaxplU6PBYMRjY0aNgLQE6
bO2p
/HJiU4Y4PkgzkEgCu0xf/mOq5DnSkX32ICoQS6jChABAe20ErPfm5t8h9YKs
Tfn9
40lAouuwD9ePRteizd4YvHtiMMwmh5PtUoCbqLefawNApAECQQD1mdBW3zL0
okUx
2pc4tttn2qArCG4CsEZMLlGRDd3FwPWJz3ZPNEEgZWXGSpA9F1QTZ6JYXIfe
jjRo
UuvRMWeBAkEA7WvzDBNcv4N+xeUKvH8ILti/BM58LraTtqJlzjQSovek0srx
tmDg
5of+xrxN6IM4p7yvQa+7YOUOukrVXjG+1QJBAI2mBrjzxgm9xTa5odn97JD7
UMFA
/WHjlMe/Nx/35V52qaav1sZbluw+TvKMcqApYj5G2SUpSNudHLDGkmd2nQEC
QFfc
lBRK8g7ZncekbGW3aRLVGVOxClnLLTzwOlamBKOUm8V6XxsMHQ6TE2D+fKJo
NUY1
2HGpk+FWwy2D1hRGuoUCQAXfaLSxtaWdPtlZTPVueF7ZikQDsVg+vtTFgpuH
loR2
6EVc1RbHHZm32yvGDY8IkcoMfJQqLONDdLfS/05yoNU=
-----END RSA PRIVATE KEY-----
```

A.4.2 Example public X.509 certificate

```
-----BEGIN CERTIFICATE-----
MIIDATCCAmqgAwIBAgIJAMB1csNuA6+kMA0GCSqGSIb3DQEBBQUAMHExCzAJ
BgNV
BAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUg
Q29t
cGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBs
ZUBl
eGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI4MzBaFw0xNDA0MDQxNTI4MzBaMIGW
MQsw
CQYDVQQGEwJVUzESMBAGA1UECBMJVGVubmVzc2VlMREwDwYDVQQHEwhGcmFu
a2xp
bjEYMBYGA1UEChMPRXhhbXBsZSBDb21wYW55MRAwDgYDVQQLEwdFeGFtcGxl
MRAw
DgYDVQQDEwdFeGFtcGxlMSIwIAYJKoZIhvcNAQkBFhNleGFtcGxlQGV4YW1w
bGUu
Y29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDjxvZXF30WL4oKjZYX
gR0Z
yaX+u3y6+JqTqiNkFa/VTnet6Ly2OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkW13
j06j
5iNn4obcCVWTL9yXNvJH+Ko+xu4Yl/ySPRrIPyTjtHdG0M2XzIlmmLqm+CAS
+KCb
JeH4tf543kIWC5pC5p3cVQIDAQABo3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG
```

```
+EIB
DQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQU
Vs3v
5afEdOeoYeVajAQE4v0WS1QwHwYDVR0jBBgwFoAUyVIc3yvra4EBz20I4BF3
9IAi
xBkwDQYJKoZIhvcNAQEFBQADgYEAgS/FF5D0Hnj44rvT6kgn3kJAvv2lj/fy
jztK
IrYS33ljXGn6gGyA4qtbXA23PrO4uc/wYCSDICDpPobh62xTCd9qObKhgwWO
i05P
SBLqUu3mwfAe15LJBJBqPVZ4K0kppePBU8m6aIZoH57L/9t4OoaL8yKs/qjK
FeI1
OFWZxvA=
-----END CERTIFICATE-----
```

Example certificate authority certificate

```
-----BEGIN CERTIFICATE-----
MIIDNzCCAqCgAwIBAgIJAMB1csNuA6+jMA0GCSqGSIb3DQEBBQUAMHExCzAJ
BgNV
BAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUg
Q29t
cGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBs
ZUBl
eGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI1NTNaFw0yMzA0MDIxNTI1NTNaMHEx
CzAJ
BgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1w
bGUg
Q29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhh
bXBs
ZUBleGFtcGxlLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA1sBn
BWPZ
0f7WJUFTJy5+01SlS5Z6DDD6Uye9vK9AycgV5B3+WC8HC5u5h91MujAC1ARP
VUOt
svPRs45qKNFIgIGRXKPAwZjawEI2sCJRSKV47i6B8bDv4WkuGvQaveZGI0ql
mN5R
1Eim2gUItRj1hgcC9rQavjlnFKDY2rlXGukCAwEAAaOB1jCB0zAdBgNVHQ4E
FgQU
yVIc3yvra4EBz20I4BF39IAixBkwgaMGA1UdIwSBmzCBmIAUyVIc3yvra4EB
z20I
4BF39IAixBmhdaRzMHExCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNz
ZWUx
GDAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEi
MCAG
CSqGSIb3DQEJARYTZXhhbXBsZUBleGFtcGxlLmNvbYIJAMB1csNuA6+jMAwG
A1Ud
EwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEADhwTebGk735yKhm8DqCxvNnE
Z0Nx
sYEYOjgRG1yXTlW5pE691fSH5AZ+T6fpwpZcWY5QYkoN6DnwjOxGkSfQC3/y
GmcU
DKBPwiZ5O2s9C+fE1kUEnrX2Xea4agVngMzR8DQ6oOauLWqehDB+g2ENWRLo
VgS+
ma5/Ycs0GTyrECY=
-----END CERTIFICATE-----
```

A.4.3 JWS Header

NOTE—Along with specifying the algorithm, the certificate chain for the signing certificate has been included.


```
{
       "alg": "RS256",
       "x5c": [

"MIIDATCCAmqgAwIBAgIJAMB1csNuA6+kMA0GCSqGSIb3DQEBBQUAMHExCzA
JBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1
wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARY
TZXhhbXBsZUBleGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI4MzBaFw0xNDA0MDQ
xNTI4MzBaMIGWMQswCQYDVQQGEwJVUzESMBAGA1UECBMJVGVubmVzc2VlMRE
wDwYDVQQHEwhGcmFua2xpbjEYMBYGA1UEChMPRXhhbXBsZSBDb21wYW55MRA
wDgYDVQQLEwdFeGFtcGxlMRAwDgYDVQQDEwdFeGFtcGxlMSIwIAYJKoZIhvc
NAQkBFhNleGFtcGxlQGV4YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4G
NADCBiQKBgQDjxvZXF30WL4oKjZYXgR0ZyaX+u3y6+JqTqiNkFa/VTnet6Ly
2OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkW13j06j5iNn4obcCVWTL9yXNvJH+Ko
+xu4Yl/ySPRrIPyTjtHdG0M2XzIlmmLqm+CAS+KCbJeH4tf543kIWC5pC5p3
cVQIDAQABo3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1N
MIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUVs3v5afEdOeoYeV
ajAQE4v0WS1QwHwYDVR0jBBgwFoAUyVIc3yvra4EBz20I4BF39IAixBkwDQY
JKoZIhvcNAQEFBQADgYEAgS/FF5D0Hnj44rvT6kgn3kJAvv2lj/fyjztKIrY
S33ljXGn6gGyA4qtbXA23PrO4uc/wYCSDICDpPobh62xTCd9qObKhgwWOi05
PSBLqUu3mwfAe15LJBJBqPVZ4K0kppePBU8m6aIZoH57L/9t4OoaL8yKs/qj
KFeI1OFWZxvA=",

"MIIDNzCCAqCgAwIBAgIJAMB1csNuA6+jMA0GCSqGSIb3DQEBBQUAMHExCzA
JBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1
wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARY
TZXhhbXBsZUBleGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI1NTNaFw0yMzA0MDI
xNTI1NTNaMHExCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDA
WBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCA
GCSqGSIb3DQEJARYTZXhhbXBsZUBleGFtcGxlLmNvbTCBnzANBgkqhkiG9w0
BAQEFAAOBjQAwgYkCgYEA1sBnBWPZ0f7WJUFTJy5+01SlS5Z6DDD6Uye9vK9
AycgV5B3+WC8HC5u5h91MujAC1ARPVUOtsvPRs45qKNFIgIGRXKPAwZjawEI
2sCJRSKV47i6B8bDv4WkuGvQaveZGI0qlmN5R1Eim2gUItRj1hgcC9rQavjl
nFKDY2rlXGukCAwEAAaOB1jCB0zAdBgNVHQ4EFgQUyVIc3yvra4EBz20I4BF
39IAixBkwgaMGA1UdIwSBmzCBmIAUyVIc3yvra4EBz20I4BF39IAixBmhdaR
zMHExCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAo
TD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb
3DQEJARYTZXhhbXBsZUBleGFtcGxlLmNvbYIJAMB1csNuA6+jMAwGA1UdEwQ
FMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEADhwTebGk735yKhm8DqCxvNnEZ0N
xsYEYOjgRG1yXTlW5pE691fSH5AZ+T6fpwpZcWY5QYkoN6DnwjOxGkSfQC3/
yGmcUDKBPwiZ5O2s9C+fE1kUEnrX2Xea4agVngMzR8DQ6oOauLWqehDB+g2E
NWRLoVgS+ma5/Ycs0GTyrECY="
    ]
}
```

A.4.4 JWS signature

ew0KICAgICJhbGciOiAiUlMyNTYiLA0KICAgICJ4NWMiOiBbDQogICAgICAg
ICJNSUlEQVRDQ0FtcWdBd0lCQWdJSkFNQjFjc051QTYra01BMEdDU3FHU0li
M0RRRUJCUVVBTUhFeEN6QUpCZ05WQkFZVEFsVlRNUkl3RUFZRFZRUUlFd2xV
Wlc1dVpYTnpaV1V4R0RBV0JnTlZCQW9URDBWNFlXMXdiR1VnUTI5dGNHRnVl
VEVRTUE0R0ExVUVBeE1IUlhoaGJYQnNaVEVpTUNBR0NTcUdTSWIzRFFFSkFS
WVRaWGhoYlhCc1pVQmxlR0Z0Y0d4bExtTnZiVEFlRncweE16QTBNRFF4TlRJ
NE16QmFGdzB4TkRBME1EUXhOVEk0TXpCYU1JR1dNUXN3Q1FZRFZRUUdFd0pW
VXpFU01CQUdBMVVFQ0JNSlZHVnVibVZ6YzJWbE1SRXdEd1lEVlFRSEV3aEdj
bUZ1YTJ4cGJqRVlNQllHQTFVRUNoTVBSWGhoYlhCc1pTQkRiMjF3WVc1NU1S
QXdEZ1lEVlFRTEV3ZEZlR0Z0Y0d4bE1SQXdEZ1lEVlFRREV3ZEZlR0Z0Y0d4
bE1TSXdJQVlKS29aSWh2Y05BUWtCRmhObGVHRnRjR3hsUUdWNFlXMXdiR1V1
WTI5dE1JR2ZNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0R05BRENCaVFLQmdRRGp4
dlpYRjMwV0w0b0tqWllYZ1IwWnlhWCt1M3k2K0pxVHFpTmtGYS9WVG5ldDZM
eTJPVDZabW1jSkVQbnEzVW5ld3BIb09RK0dmaGhUa1cxM2owNmo1aU5uNG9i
Y0NWV1RMOXlYTnZKSCtLbyt4dTRZbC95U1BScklQeVRqdEhkRzBNMlh6SWxt
bUxxbStDQVMrS0NiSmVINHRmNTQza0lXQzVwQzVwM2NWUUlEQVFBQm8zc3dl
VEFKQmdOVkhSTUVBakFBTUN3R0NXQ0dTQUdHK0VJQkRRUWZGaDFQY0dWdVUx
Tk1JRWRsYm1WeVlYUmxaQ0JEWlhKMGFXWnBZMkYwWlRBZEJnTlZIUTRFRmdR
VVZzM3Y1YWZFZE9lb1llVmFqQVFFNHYwV1MxUXdId1lEVlIwakJCZ3dGb0FV
eVZJYzN5dnJhNEVCejIwSTRCRjM5SUFpeEJrd0RRWUpLb1pJaHZjTkFRUZC
UUFEZ1lFQWdTL0ZGNUQwSG5qNDRydlQ2a2duM2tKQXZ2MmxqL2Z5anp0S0ly
WVMzM2xqWEduNmdHeUE0cXRiWEEyM1ByTzR1Yy93WUNTRElDRHBQb2JoNjJ4
VENkOXFPYktoZ3dXT2kwNVBTQkxxVXUzbXdmQWUxNUxKQkpCcVBWWjRLMGtw
cGVQQlU4bTZhSVpvSDU3TC85dDRPb2FMOHlLcy9xaktGZUkxT0ZXWnh2QT0i
LA0KICAgICAgICAiTUlJRE56Q0NBcUNnQXdJQkFnSUpBTUIxY3NOdUE2K2pN
QTBHQ1NxR1NJYjNEUUVCQlFVQU1IRXhDekFKQmdOVkJBWVRBbFZUTVJJd0VB
WURWUVFJRXdsVVpXNXVaWE56WldVeEdEQVdCZ05WQkFvVEQwVjRZVzF3YkdV
Z1EyOXRjR0Z1ZVRFUU1BNEdBMVVFQXhNSFJYaGhiWEJzWlRFaU1DQUdDU3FH
U0liM0RRRUpBUllUWlhoaGJYQnNaVUJsZUdGdGNHeGxMbU52YlRBZUZ3MHhN
ekEwTURReE5USTFOVE5hRncweU16QTBNREl4TlRJMU5UTmFNSEV4Q3pBSkJn
TlZCQVlUQWxWVE1SSXdFQVlEVlFRSUV3bFVaVzV1WlhOelpXVXhHREFXQmdO
VkJBb1REMFY0WVcxd2JHVWdRMjl0Y0dGdWVURVFNQTRHQTFVRUF4TUhSWGho
YlhCc1pURWlNQ0FHQ1NxR1NJYjNEUUVKQVJZVFpYaGhiWEJzWlVCbGVHRnRj
R3hsTG1OdmJUQ0JuekFOQmdrcWhraUc5dzBCQVFFRkFBT0JqUUF3Z1lrQ2dZ
RUExc0JuQldQWjBmN1dKVUZUSnk1KzAxU2xTNVo2REReNlV5ZTl2SzlBeWNn
VjVCMytXQzhIQzV1NWg5MU11akFDMUFSUFZVT3RzdlBSczQ1cUtORklnSUdS
WEtQQXdaamF3RUkyc0NKUlNLVjQ3aTZCOGJEdjRXa3VHdlFhdmVaR0kwcWxt
TjVSMUVpbTJnVUl0UmoxaGdjQzlyUWF2amxuRktEWTJybFhHdWtDQXdFQUFh
T0IxakNCMHpBZEJnTlZIUTRFRmdRVXlWSWMzeXZyYTRFQnoyMEk0QkYzOUlB
aXhCa3dnYU1HQTFVZEl3U0JtekNCbUlBVXlWSWMzeXZyYTRFQnoyMEk0QkYz
OUlBaXhCbWhkYVJ6TUhFeEN6QUpCZ05WQkFZVEFsVlRNUkl3RUFZRFZRUUlF
d2xVWlc1dVpYTnpaV1V4R0RBV0JnTlZCQW9URDBWNFlXMXdiR1VnUTI5dGNH
RnVlVEVRTUE0R0ExVUVBeE1IUlhoaGJYQnNaVEVpTUNBR0NTcUdTSWIzRFFF
SkFSWVRaWGhoYlhCc1pVQmxlR0Z0Y0d4bExtTnZiWUlKQU1CMWNzTnVBNitq
TUF3R0ExVWRFd1FGTUFNQkFmOHdEUVlKS29aSWh2Y05BUUVGQlFBRGdZRUFE
aHdUZWJHazczNXlLaG04RHFDeHZObkVaME54c1lFWU9qZ1JHMXlYVGxXNXBF
NjkxZlNINUFaK1Q2ZnB3cFpjV1k1UVlrb042RG53ak94R2tTZlFDMy95R21j
VURLQlB3aVo1TzJzOUMrZkUxa1VFbnJYMlhlYTRhZ1ZuZ016UjhEUTZvT2F1
TFdxZWhEQitnMkVOV1JMb1ZnUyttYTUvWWNzMEdUeXJFQ1k9Ig0KICAgIF0N
Cn0.ew0KICAgICJ2ZXJzaW9uIjogIjEuMC4wIiwNCiAgICAiaWQiOiAiMzNj
ZmY0MTYtZTMzMS00YzlkLTk2OWUtNTM3M2ExNzU2MTIwIiwNCiAgICAiYWN0
b3IiOiB7DQogICAgICAgICJtYm94IjogIm1haWx0bzpleGFtcGxlQGV4YW1w

bGUuY29tIiwNCiAgICAgICAgIm5hbWUiOiAiRXhhbXBsZSBMZWFybmVyIiwN
CiAgICAgICAgIm9iamVjdFR5cGUiOiAiQWdlbnQiDQogICAgfSwNCiAgICAi
dmVyYiI6IHsNCiAgICAgICAgImlkIjogImh0dHA6Ly9hZGxuZXQuZ292L2V4
cGFwaS92ZXJicy9leHBlcmllbmNlZCIsDQogICAgICAgICJkaXNwbGF5Ijog
ew0KICAgICAgICAgICAgImVuLVVTIjogImV4cGVyaWVuY2VkIg0KICAgICAg
ICB9DQogICAgfSwNCiAgICAib2JqZWN0Ijogew0KICAgICAgICAiaWQiOiAi
aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g_dj14aDRrSWlIM1NtOCIs
DQogICAgICAgICJvYmplY3RUeXBlIjogIkFjdGl2aXR5IiwNCiAgICAgICAg
ImRlZmluaXRpb24iOiB7DQogICAgICAgICAgICAibmFtZSI6IHsNCiAgICAg
ICAgICAgICAgICAiZW4tVVMiOiAiVGF4IFRpcHMgJiBJbmZvcmlhdGlvbiA6
IEhvdyB0byBGaWxlIGEgVGF4IFJldHVybiAiDQogICAgICAgICAgICB9LA0K
ICAgICAgICAgICAgImRlc2NyaXB0aW9uIjogew0KICAgICAgICAgICAgICAg
ICJlbi1VUyI6ICJGaWxpbmcgYSB0YXggcmV0dXJuIHdpbGwgcmVxdWlyZSBm
aWxsaW5nIG91dCBlaXRoZXIgYSAxMDQwLCAxMDQwQSBvciAxMDQwRVogZm9y
bSINCiAgICAgICAgICAgIH0NCiAgICAgICAgfQ0KICAgIH0sDQogICAgInRp
bWVzdGFtcCI6ICIyMDEzLTA0LTAxVDEyOjAwOjAwWiINCn0.FWuwaPhwUbkk
7h9sKW5zSvjsYNugvxJ-
TrVaEgt_ DCUT0bmKhQScRrjMB6P9O50uznPwT66oF1NnU_G0HVhRzS5voiXE
-y7tT3z0M3-8A6YK009Bk_digVUul-
HA4Fpd5IjoBBGe3yzaQ2ZvzarvRuipvNEQCI0onpfuZZJQ0d8

A.4.5 Signed Statement

```
{
       "version": "1.0.0",
       "id": "33cff416-e331-4c9d-969e-5373a1756120",
       "actor": {
              "mbox": "mailto:example@example.com",
              "name": "Example Learner",
              "objectType": "Agent"
       },
       "verb": {
              "id": "http://adlnet.gov/expapi/verbs/experienced",
              "display": {
                     "en-US": "experienced"
              }
       },
       "object": {
              "id": "https://www.youtube.com/watch?v=xh4kIiH3Sm8",
              "objectType": "Activity",
              "definition": {
                     "name": {
                            "en-US": "Tax Tips & Information : How to
File a Tax Return "
                     },
                     "description": {
                            "en-US": "Filing a tax return will require
filling out either a 1040, 1040A or 1040EZ form"
                     }
              }
       },
       "timestamp": "2013-04-01T12:00:00Z",
       "attachments": [
              {
```

```
                    "usageType":
"http://adlnet.gov/expapi/attachments/signature",
                    "display": { "en-US": "Signature" },
                    "description": { "en-US": "A test signature" },
                    "contentType": "application/octet-stream",
                    "length": 4235,
                    "sha2":
"672fa5fa658017f1b72d65036f13379c6ab05d4ab3b6664908d8acf0b6a
0c634"
            }
      ]
}
```

IEEE SA
STANDARDS ASSOCIATION

RAISING THE WORLD'S STANDARDS

Connect with us on:

Twitter: twitter.com/ieeesa
Facebook: facebook.com/ieeesa
LinkedIn: linkedin.com/groups/1791118
Beyond Standards blog: beyondstandards.ieee.org
YouTube: youtube.com/ieeesa

standards.ieee.org
Phone: +1 732 981 0060

IEEE