



International
Standard

**ISO/IEC/IEEE
39274-1-1**

First edition
2025-10

**Learning technology — JavaScript
Object Notation (JSON) data model
format and Representational State
Transfer (RESTful) web service for
learner experience data tracking
and access —**

**Part 1-1:
xAPI using JSON serialization and
RESTful data transport**

*Technologie de l'apprentissage — Format de modèle de
données JavaScript Object Notation (JSON) et service web
Representational State Transfer (RESTful) pour le suivi des
données de l'expérience de l'apprenant et l'accès à ces données —*

*Partie 1-1: xAPI utilisant la sérialisation JSON et le transport de
données RESTful*

Reference number
ISO/IEC/IEEE 39274-1-1:2025(en)

© IEEE 2023

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25, 2026 at 13:35:19 UTC from IEEE Xplore. Restrictions apply.

Full document details: [ISO/IEC/IEEE 39274-1-1-2025](#)



COPYRIGHT PROTECTED DOCUMENT

© IEEE 2023

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from IEEE at the address below.

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

Email: stds.ipr@ieee.org
Website: www.ieee.org

Published in Switzerland

© IEEE 2023 – All rights reserved

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25, 2026 at 13:35:19 UTC from IEEE Xplore. Restrictions apply.

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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.

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and <https://patents.iec.ch>. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

ISO/IEC/IEEE 39274-1-1 was prepared by the LAN/MAN of the IEEE Computer Society (as IEEE Std 9274.1.1™-2023) and drafted in accordance with its editorial rules. It was adopted, under the “fast-track procedure” defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 36, *Information technology for learning, education and training*.

A list of all parts in the ISO/IEC/IEEE 39274 series can be found on the ISO and IEC websites.

ISO/IEC/IEEE 39274-1-1:2025(en)

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

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

ISO/IEC/IEEE 39274-1-1:2025(en)

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

Copyright © 2023 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 6 October 2023. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 979-8-8557-0031-2 STD26400

Print: ISBN 979-8-8557-0032-9 STDPD26400

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

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

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>.

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.

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.

³ Available at: <https://ieeexplore.ieee.org/browse/standards/collection/ieee>.

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 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.

⁴ Available at: <https://standards.ieee.org/standard/index.html>.

⁵ Available at: <https://standards.ieee.org/about/sasb/patcom/materials.html>.

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, Technical Editor*

Nick Washburn, *Secretary*

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

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	Joerg Boeselt	Mark Davis
Catherine Allman	Jeremy Brockman	Jhorlin De Armas
Jonathan Archibald	Jennifer Cameron	Andrew Downes
Keith Bartolotta	Rob Chadwick	Andriy Drozdyuk
Steve Baumgartner	Rich Chetwynd	Brian K. Duck
Al Bejcek	Lewis Cowper	Russell Duhon
Marcus Birtwhistle	Ingo Dahn	David Ells

ISO/IEC/IEEE 39274-1-1:2025(en)

Paul Esch	Dan Kuemmel	Kris Rockwell
Michael Flores	Richard Lenz	Jennifer Rogers
Steve Flowers	Jason Lewis	Mike Rustici
Richard Fouchaux	Robert Lowe	Chris Sawwa
Walt Grata	Zach Lowry	Matteo Scaramuccia
Joe Gorup	Bill McDonald	Ángel Serrano
Doug Hagy	Brian J. Miller	Ali Shahrazad
Chaim-Leib Halbert	Kris Miller	Ryan Smith
Dennis Hall	Fiona Leteney	Roger Swetnam
Chris Handorf	Tim Martin	Greg Tatka
Luke Hickey	Dave Mozealous	Stephen Trevorrow
Thomas Ho	Tyler Mulligan	Chad Udell
Lang Holloman	Mike Palmer	Anto Valan
Nikolaus Hruska	David Pate	Melanie VanHorn
Loic Jeannin	Danny Pham	Yannick Warnier
David N. Johnson	Jeff Place	Michael Wheeler
Eric Johnson	Rick Raymer	Andy Whitaker
Patrick Kedziora	Michael Roberts	Lou Wolford
John Kleeman	Paul Roberts	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	James Goodell	Linda Steedman
Charles Barest	Werner Hoelzl	Andreas Tsouchlaris
Avron Barr	William Hoyt	Lisa Ward
Shelly Blake-Plock	Andy Johnson	Karl Weber
Thomas Creighton	Piotr Karocki	Yu Yuan
David Fuschi	Lakshman Raut	Janusz Zalewski

ISO/IEC/IEEE 39274-1-1:2025(en)

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

- 1. Overview
 - 1.1 Scope
 - 1.2 Purpose
 - 1.3 Word usage
- 2. Normative references
- 3. Definitions, acronyms, and abbreviations
 - 3.1 Definitions
 - 3.2 Acronyms and abbreviations
- 4. Learning Record Stores (LRSs)
 - 4.1 LRS Communication
 - 4.2 LRS Data Requirements
- 5. Content—Learning Record Providers (LRPs) and Learning Record Consumers (LRCs)
 - 5.1 LRP and LRC Communication
 - 5.2 LRP Data Requirements
- Annex A (informative) **xAPI Base Standard Examples**
 - A.1 Example Statements
 - A.2 Examples of Statement's Objects of different types
 - A.3 Example definitions for Activities of type cmi.interaction
 - A.4 Example Signed Statement

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.

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.

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.

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.

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.

¹⁵ Information on references can be found in Clause 2.

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.

ISO/IEC/IEEE 39274-1-1:2025(en)

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

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

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

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.

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.

ISO/IEC/IEEE 39274-1-1:2025(en)

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

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
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

ISO/IEC/IEEE 39274-1-1:2025(en)

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

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	<p>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.</p> <p>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.</p> <p>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.</p>	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* still 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

ISO/IEC/IEEE 39274-1-1:2025(en)

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

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 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) <i>shall</i> 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

ISO/IEC/IEEE 39274-1-1:2025(en)

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

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