

# Mục lục

<b>1</b>	<b>Experience API</b>	<b>1</b>
1.1	Advanced Distributed Learning (ADL) Co-Laboratories	1
1.1.1	Bản tiếng Việt 2014.8.1	1
1.1.2	”Dự án Tin Can”	1
1.1.3	Thành viên biên soạn bản tiếng Nhật của ”Dự án Tin Can” (theo thứ tự bảng chữ cái tiếng Nhật)	2
1.2	Mục lục	2
1.2.1	2.1 Vị trí của ADL trong xAPI	5
1.2.2	2.2 Những người đóng góp	5
1.2.3	2.2.1 Thành viên nhóm công tác	5
1.2.4	2.3 Hướng dẫn đọc cho người không chuyên kỹ thuật	5
1.3	4.0 Statement	8
1.3.1	4.1 Thuộc tính của Statement	8
1.3.2	4.2 Truy xuất Statement	26
1.4	5.0 Các kiểu dữ liệu khác	28
1.4.1	5.1 Tài liệu	28
1.4.2	5.2 Bản đồ ngôn ngữ	29
1.4.3	5.3 Mở rộng	29
1.4.4	5.4 Siêu dữ liệu định danh	29
1.5	6.0 Giao tiếp lúc chạy	30
1.5.1	6.1 Mã hóa	30
1.5.2	6.2 Quản lý phiên bản API	30
1.5.3	6.3 Đồng thời (Concurrency)	31
1.5.4	6.4 Bảo mật	32
1.6	7.0 Truyền dữ liệu (REST)	34
1.6.1	7.1 Mã lỗi	34
1.6.2	7.2 Statement API	36
1.6.3	7.3 Document API	38
1.6.4	7.4 State API	39
1.6.5	7.5 Activity Profile API	40
1.6.6	7.6 Agent Profile API	40
1.6.7	7.7 Tài nguyên About	42
1.6.8	7.8 Yêu cầu Cross-Origin	42
1.6.9	7.9 Validation	43
1.6.10	7.10 HTTP HEAD	43
1.6.11	Endpoint bắt buộc	61
1.6.12	OAuth Endpoint	61

## 1 Experience API

### 1.1 Advanced Distributed Learning (ADL) Co-Laboratories

#### 1.1.1 Bản tiếng Việt 2014.8.1

#### 1.1.2 ”Dự án Tin Can”

Tổ chức phi lợi nhuận Japan e-Learning Consortium và Mobile Learning Consortium đã cùng thành lập ”Dự án Tin Can”. Tài liệu này là bản dịch tiếng Nhật của Experience API (v1.0.1) do các thành viên ”Dự án Tin Can”

thực hiện.

Vui lòng gửi các câu hỏi về tài liệu này tới:

tincan@elc.or.jp

### 1.1.3 Thành viên biên soạn bản tiếng Nhật của ”Dự án Tin Can” (theo thứ tự bảng chữ cái tiếng Nhật)

伊藤彰孝 (株式会社 ベネッセコーポレーション)

勝田祥生 (株式会社レビックグローバル)

加藤泰久 (NTT ラーニングシステムズ株式会社)

熊澤 剛 (株式会社ヒューマンサイエンス)

佐伯秀雄 (NTT ラーニングシステムズ株式会社)

田中頼人 (早稲田大学)

徳弘太郎 (ヤマハ株式会社)

藤井直人 (TinCan プロジェクト・リーダー 株式会社ヌーサイト)

星野忠明 (エスエイティーティー株式会社)

程田恭介 (株式会社富士通ラーニングメディア)

前田 宏 (株式会社ジンジャーアップ)

Copyright 2013 Advanced Distributed Learning (ADL) Initiative, U.S. Department of Defence

Tài liệu này được cấp phép theo Apache License, Version 2.0 (sau đây gọi là ”Giấy phép”). Bạn không được sử dụng tài liệu này nếu không tuân thủ Giấy phép. Bạn có thể nhận bản sao Giấy phép tại: <http://www.apache.org/licenses/LICENSE-2.0>

Trừ khi pháp luật hiện hành yêu cầu hoặc có thỏa thuận bằng văn bản, phần mềm được cung cấp theo Giấy phép được phân phối trên cơ sở ”as-is” (nguyên trạng), không có bất kỳ bảo đảm hay điều kiện nào, dù được nêu rõ hay hàm ý. Vui lòng tham khảo nội dung trong Giấy phép để biết các quyền được cho phép và các giới hạn áp dụng.

Tài liệu này được biên soạn bởi các thành viên Nhóm Công tác Experience API (xem bảng tại 2.2.1), với sự hỗ trợ của sáng kiến Advanced Distributed Learning (ADL) thuộc Văn phòng Trợ lý Bộ trưởng Quốc phòng (phụ trách sẵn sàng tác chiến). Vui lòng gửi phản hồi hoặc câu hỏi tới [helpdesk@adlnet.gov](mailto:helpdesk@adlnet.gov).

## 1.2 Mục lục

- 1.0. Lịch sử sửa đổi
- 2.0. Vai trò của Experience API
  - 2.1. Vị trí của ADL trong xAPI
  - 2.2. Những người đóng góp
    - \* 2.2.1. Thành viên nhóm công tác
    - \* 2.2.2. Những người đóng góp cho việc thu thập yêu cầu
  - 2.3 Hướng dẫn đọc cho người không chuyên kỹ thuật
- 3.0. Định nghĩa thuật ngữ
- 4.0. Statement
  - 4.1. Các thuộc tính của Statement

- \* 4.1.1. id (định danh)
  - \* 4.1.2. Actor (chủ thể)
  - \* 4.1.3. Verb (động từ)
  - \* 4.1.4. Object (đối tượng)
  - \* 4.1.5. Result (kết quả)
  - \* 4.1.6. Context (ngữ cảnh)
  - \* 4.1.7. Timestamp (mốc thời gian)
  - \* 4.1.8. Stored
  - \* 4.1.9. Authority (thẩm quyền)
  - \* 4.1.10. Version (phiên bản)
  - \* 4.1.11. Attachments (tệp đính kèm)
  - \* 4.1.12. Ràng buộc dữ liệu
- 4.2. Truy xuất Statement
- 4.3. Vô hiệu hóa
- 4.4. Statement có chữ ký
- 5.0. Các kiểu khác
  - 5.1. Tài liệu
  - 5.2. Bản đồ ngôn ngữ
  - 5.3. Mở rộng
  - 5.4. Siêu dữ liệu định danh
- 6.0. Giao tiếp thời gian chạy
  - 6.1. Mã hóa
  - 6.2. Quản lý phiên bản API
  - 6.3. Đồng thời
  - 6.4. Bảo mật
    - \* 6.4.1. Cách xử lý từng kịch bản
    - \* 6.4.2. Phạm vi xác thực OAuth
- 7.0. Truyền dữ liệu (REST)
  - 7.1. Mã lỗi
  - 7.2. API Statement
    - \* 7.2.1 PUT Statement
    - \* 7.2.2 POST Statement
    - \* 7.2.3 GET Statement
    - \* 7.2.4 Statement đã bị vô hiệu hóa
  - 7.3. API Tài liệu
  - 7.4. API State
  - 7.5. API Hồ sơ hoạt động
  - 7.6. API Hồ sơ tác nhân
  - 7.7. Tài nguyên About
  - 7.8. Yêu cầu cross-origin
  - 7.9. Validation
  - 7.10. HTTP HEAD
- Phụ lục A: Ví dụ Statement
- Phụ lục B: Ví dụ các loại đối tượng Statement khác nhau
- Phụ lục C: Ví dụ định nghĩa cho loại hoạt động ”cml.interaction”
- Phụ lục D: Chuyển đổi sang Statement định dạng 1.0.0
- Phụ lục E: Ví dụ Statement có chữ ký
- Phụ lục F: Bảng tất cả endpoint
- Phụ lục G: Ví dụ cross-domain request

Rustici Software, đơn vị cung cấp Project Tin Can API, đã thực hiện các sửa đổi đối với API trước cuộc họp khởi động vào tháng 4 năm 2012. Các sửa đổi này đã được biểu quyết tại cuộc họp và được đưa vào đặc tả để trở thành phiên bản 0.9.

Từ 0.90 đến 0.95 (2012/08/31)

Các động từ và loại hoạt động được xem là "cốt lõi" đã bị loại khỏi đặc tả. Đồng thời, các tham chiếu tới các động từ đó trong phần kết quả, ngữ cảnh, tương tác và định nghĩa hoạt động cũng bị loại bỏ. Thay vì tự định nghĩa động từ, nhà triển khai được khuyến nghị sử dụng các động từ do cộng đồng định nghĩa.

- Sử dụng URI cho động từ, loại hoạt động và khóa mở rộng.
- Rà soát phạm vi và một số chi tiết triển khai, đồng thời bổ sung giải thích.
- Với Agent, thay đổi từ góc nhìn cá nhân sang góc nhìn persona.
- Bỏ yêu cầu hợp nhất Agent theo Friend of a Friend (FOAF).
- Thay đổi để đối tượng Agent có đúng một thuộc tính ID duy nhất (thay vì một hoặc nhiều thuộc tính ID duy nhất).

Từ 0.95 đến 1.0.0 (2013/4/26)

Đã thực hiện nhiều cải tiến và làm rõ như sau:

- Bổ sung khái niệm tài liệu đính kèm.
- Thay đổi siêu dữ liệu hoạt động từ XML sang JSON.
- Cập nhật liên quan đến việc vô hiệu hóa Statement.
- Bổ sung API tài liệu.
- Cập nhật về truy xuất trong API Statement.
- Bổ sung Statement có chữ ký.

0.95...1.0.0

Từ 1.0.0 đến 1.0.1 (2013/10/01)

Bổ sung các nội dung làm rõ và ví dụ, bao gồm:

- Sửa nhiều lỗi chính tả.
- Bổ sung tài liệu tham khảo.

1.0.0...1.0.1

Từ 1.0.1 đến 1.0.2 (2014/10/01)

- Bổ sung thông tin tùy chọn/bắt buộc vào bảng.
- Bổ sung tiêu đề bảng còn thiếu trong các thành phần tương tác.
- Thay đổi yes/no của tệp đính kèm thành bắt buộc/tùy chọn.
- Làm rõ mục đích của phần tử `moreInfo`.

1.0.1...1.0.2

xAPI cung cấp:

- Cấu trúc và định nghĩa của Statement, trạng thái, người học, hoạt động và đối tượng như phương tiện để Activity Provider truyền tải trải nghiệm.
- Cơ chế truyền dữ liệu để ghi và truy xuất (không bao gồm xác thực nội dung) các đối tượng nêu trên đối với LRS. Lưu ý rằng hệ thống ghi hoặc truy xuất thông tin không nhất thiết phải là Activity Provider. LRS có thể giao tiếp với LRS khác hoặc hệ thống báo cáo.
- Cơ chế bảo mật cho phép truyền thông tin đáng tin cậy giữa LRS và nguồn dữ liệu.

xAPI là điểm khởi đầu cho một loạt đề xuất kỹ thuật nhằm cung cấp kiến trúc nâng cao cho học tập và đào tạo trực tuyến. Ví dụ áp dụng xAPI gồm dịch vụ xác thực, dịch vụ tìm kiếm, dịch vụ trực quan hóa và dịch vụ dữ liệu cá nhân. Đặc tả này không nêu chi tiết triển khai các dịch vụ đó, nhưng xAPI được thiết kế với tiền đề là một tư tưởng kiến trúc lớn hơn.

### 1.2.1 2.1 Vị trí của ADL trong xAPI

Sáng kiến Advanced Distributed Learning (ADL) đảm nhiệm vai trò văn phòng điều phối và điều phối viên trong quá trình phát triển xAPI. xAPI được định vị là một phần trong kiến trúc đào tạo và học tập của ADL, hướng tới việc cho phép học ở mọi lúc mọi nơi. ADL xem xAPI không chỉ là phần mở rộng của SCORM (Sharable Content Object Reference Model) để hỗ trợ các trường hợp sử dụng tương tự, mà còn là cách hỗ trợ các trường hợp sử dụng mà SCORM không thể đáp ứng, do ADL và cộng đồng học tập phân tán đề xuất.

### 1.2.2 2.2 Những người đóng góp

*Xin cảm ơn tất cả những người đã đóng góp cho dự án Experience API. Nhiều người đã tham gia các cuộc họp hằng tuần và hỗ trợ hoàn thiện đặc tả này để nó hữu ích cho toàn bộ cộng đồng học tập phân tán. Ngoài ra, nhiều người đã cung cấp mã mẫu, sản phẩm và tài liệu để hỗ trợ nhóm biên soạn và chỉnh sửa đặc tả. Chúng tôi cũng xin cảm ơn những người đã chia sẻ thông tin hữu ích và thẳng thắn về việc sử dụng SCORM cũng như các thực hành tốt nhất liên quan đến học tập trong tổ chức của mình. Dựa trên các trường hợp sử dụng, kinh nghiệm và kiến thức mà mọi người cung cấp, ADL và cộng đồng đã có thể định nghĩa rõ ràng bước đầu tiên của kiến trúc đào tạo và học tập - Experience API. Chính các bạn là những lãnh đạo cộng đồng mà chúng tôi có thể tin cậy để cung cấp hoạt động đào tạo và giáo dục tốt nhất.*

Kristy S. Murray, Ed.D. Director, ADL Initiative OSD, Training Readiness & Strategy (TRS)

### 1.2.3 2.2.1 Thành viên nhóm công tác

**1.2.3.1 2.2.2 Người đóng góp cho việc thu thập yêu cầu** Trong quá trình thu thập yêu cầu cho xAPI, chúng tôi đã nhận được phản hồi riêng lẻ từ rất nhiều cá nhân và tổ chức về SCORM®, học tập phân tán và công nghệ học tập nói chung. Không thể liệt kê đầy đủ tất cả, nhưng các nguồn thông tin quan trọng để tổng hợp đặc tả xAPI gồm có: white paper do nhóm Learning Education Training Systems Interoperability (LESTI) tổng hợp năm 2008, trang web *UserVoice* của Rustici Software, các cuộc phỏng vấn riêng lẻ, và nhiều bài viết blog khác nhau.

### 1.2.4 2.3 Hướng dẫn đọc cho người không chuyên kỹ thuật

Tài liệu này là tài liệu chính thức mô tả cách triển khai xAPI cho nhiều hệ thống khác nhau. Đồng thời, đây cũng là tài liệu kỹ thuật hướng tới cá nhân và tổ chức triển khai công nghệ này, với mục tiêu giúp nhà triển khai có thể phát triển các công cụ, hệ thống và dịch vụ độc lập nhưng vẫn có khả năng tương tác.

Do nhiều công cụ, hệ thống và dịch vụ sẽ được thiết kế dựa trên đặc tả định nghĩa tại đây, câu chữ và hình thức trong tài liệu này được xây dựng để, trong mức tối đa có thể, vẫn *đễ tiếp cận* với người không chuyên kỹ thuật. Vì vậy, phần *mô tả tổng quan* của từng phần trong xAPI được gắn tiêu đề **Giải thích** hoặc **Bối cảnh**. Các phần kỹ thuật được gắn tiêu đề **Yêu cầu**, **Chi tiết**, hoặc **Ví dụ**.

Nói khái quát, nếu một đoạn trông có tính kỹ thuật hoặc giống đặc tả yêu cầu, hãy diễn giải nó theo hướng đó. Điều này đặc biệt áp dụng cho các đoạn dài, mô tả chi tiết hơn hoặc các bảng biểu, nơi trình bày nhiều yêu cầu theo cách không trực quan.

- Activity (hoạt động)

- Activity Provider (nhà cung cấp hoạt động, AP)
- Actor (chủ thể)
- Xác thực (Authentication)
- Ủy quyền (Authorization)
- Base Endpoint
- Client
- Cộng đồng thực hành
- Thuật ngữ kiểm soát
- Experience API (xAPI)
- Bất biến (Immutable)
- IRI (Internationalized Resource Identifier)
- IRL (Internationalized Resource Locator)
- Inverse Functional Identifier
- LMS
- LRS
- MUST / SHOULD / MAY
- Hồ sơ (Profile)
- Đăng ký (Registration)
- REST (REpresentational State Transfer)
- Dịch vụ
- Statement
- Tin Can API (TCAPI)
- Verb (động từ)

**Activity (hoạt động):** Activity là một dạng của tân ngữ, tương ứng với phần "cái này" trong câu "Tôi đã làm cái này". Đó là một đối tượng nào đó mà Actor đã tương tác. Khi kết hợp có ý nghĩa với Verb, nó có thể trở thành đơn vị ghi nhận "chỉ dẫn, trải nghiệm, hoặc kết quả". Khái niệm activity có thể được hiểu theo phạm vi rộng và trong một số trường hợp có thể chỉ vật thể cụ thể như chiếc ghế (cả vật lý lẫn ảo). Trong Statement "Anna đã thử công thức bánh", "công thức" là một Activity trong Statement xAPI. Các ví dụ khác về Activity gồm "sách", "khóa học e-learning", "đi bộ đường dài", "cuộc họp".

**Activity Provider (nhà cung cấp hoạt động):** Đối tượng phần mềm giao tiếp với LRS và ghi nhận trải nghiệm học tập. Khái niệm này giống với gói SCORM tập hợp học liệu và các đối tượng có thể giao tiếp, nhưng nhà cung cấp hoạt động có thể tách rời khỏi chính trải nghiệm mà nó định truyền đạt.

**Actor:** Danh tính hoặc khía cạnh bên ngoài của một cá nhân hay một nhóm. Actor được ghi lại trong Statement khi thực hiện hành động trong một hoạt động.

**Xác thực (Authentication):** Việc xác minh danh tính của người dùng hoặc hệ thống. Xác thực cho phép trao đổi giữa hai chủ thể "được tin cậy".

**Ủy quyền (Authorization):** Việc cấp quyền sử dụng nào đó theo vai trò của người dùng hoặc hệ thống. Đây là quá trình khiến một người dùng hoặc hệ thống được người khác tin cậy.

**Base Endpoint:** Đường dẫn dài nhất dùng chung cho tất cả endpoint của xAPI, bao gồm dấu / ở cuối. Ví dụ: với endpoint Statement là `http://example.com/xAPI/statements`, Base Endpoint là `http://example.com/xAPI/`.

**Client:** Mọi thực thể có thể tương tác với LRS. Client có thể là nhà cung cấp hoạt động, công cụ báo cáo, LMS hoặc LRS khác.

**Cộng đồng thực hành:** Nhóm thường được liên kết bởi động cơ, vai trò hoặc mục tiêu chung. Nhóm này cũng hành động theo các cách thức chung.

**Hồ sơ cộng đồng:** Bộ tài liệu riêng tổng hợp cách nhìn nhận một hoạt động học tập cụ thể khi sử dụng xAPI, bao gồm use case của cộng đồng thực hành, thuật ngữ kiểm soát, đặc tả triển khai và ví dụ định nghĩa sẵn (ví dụ: công thức).

**Thuật ngữ kiểm soát:** Danh sách thuật ngữ bị giới hạn và đã được đồng thuận, do cộng đồng thực hành xây dựng. Mục đích của thuật ngữ kiểm soát là đảm bảo tính nhất quán trong biểu đạt ý nghĩa, lưu trữ và truy xuất dữ liệu xAPI. Danh sách này cần được quản lý vì các thuật ngữ được dự kiến sử dụng trong các lĩnh vực vấn đề hoặc miền cụ thể. Bản thân việc bảo trì danh sách cũng cần được quản lý. Danh sách thuật ngữ sẽ tiếp tục được mở rộng nhưng phải tuân theo chính sách của cộng đồng thực hành.

**Experience API (xAPI):** API được quy định trong tài liệu này, là một sản phẩm của dự án Tin Can. Đây là phương thức đơn giản, gọn nhẹ để các Actor được cấp quyền lưu trữ và truy xuất "bản ghi học tập có thể mở rộng, hồ sơ người học và hồ sơ trải nghiệm học tập". Nó cũng độc lập với nền tảng.

**Bất biến:** Tính từ mô tả sự kiện không thay đổi. Ngoại trừ một số trường hợp, Statement của xAPI là bất biến. Tính bất biến đảm bảo giữ nguyên tính đồng nhất giữa các Statement được sao chép khi chúng được chia sẻ giữa các LRS.

**IRI (Internationalized Resource Identifiers):** Định danh duy nhất, có thể là IRL. Trong xAPI, mọi IRI nên là IRI tuyệt đối đầy đủ có chứa scheme. Không nên dùng IRI tương đối. IRL nên được định nghĩa trong miền do bên tạo IRL kiểm soát.

**IRI (Internationalized Resource Identifiers):** Là định danh duy nhất, đôi khi cũng là IRL. IRI được dùng để xác định các tân ngữ như Verb, Activity hoặc loại Activity. Khác với URI, IRI có thể sử dụng ký tự ngoài ASCII để hỗ trợ ngôn ngữ quốc tế.

IRI luôn bao gồm scheme. Đây không phải yêu cầu riêng của tiêu chuẩn này mà theo định nghĩa IRI trong RFC 3987. Cái gọi là "IRI tương đối" thực chất không phải IRI.

**Internationalized Resource Locator (IRL):** Trong đặc tả này, khi URL có thể thay thế bằng URI thì IRL cũng có thể thay thế bằng IRI. (Quy tắc thay thế URL -> URI và IRL -> IRI là giống nhau.) Một số cộng đồng thực hành dùng IRL đôi khi cũng chỉ dùng URL, kể cả khi cách đó có thể không phù hợp về mặt kỹ thuật trong phạm vi xAPI.

**Inverse Functional Identifier:** Định danh duy nhất cho một cá nhân hoặc một nhóm cụ thể. Được dùng để xác định Actor hoặc Group.

**Learning Management System (LMS):** Theo định nghĩa của Learning Systems Architecture Lab (ghi chú dịch: Đại học Carnegie Mellon), LMS là gói phần mềm để cung cấp một hoặc nhiều khóa học cho một hoặc nhiều người học. LMS nói chung là hệ thống dựa trên web để xác thực người học, đăng ký khóa học, hoàn tất khóa học và đánh giá. Trong đặc tả này, thuật ngữ LMS được dùng trong ngữ cảnh hệ thống hiện có triển khai tiêu chuẩn.

**Learning Record Store (LRS):** Hệ thống lưu trữ thông tin về học tập. Trước khi xAPI xuất hiện, phần lớn LRS thực chất là LMS. Tuy nhiên trong đặc tả này, khi dùng thuật ngữ LRS, chúng tôi muốn nhấn mạnh rằng không nhất thiết cần LMS đầy đủ chức năng để triển khai xAPI. Để thực hiện chức năng của mình, xAPI cần LRS (không nhất thiết là LMS).

**MUST / SHOULD / MAY:** Ba mức cam kết về tính phù hợp với đặc tả xAPI. Hệ thống không đáp ứng điều kiện MUST (hoặc MUST NOT) thì không phù hợp xAPI. Hệ thống không đáp ứng điều kiện SHOULD không vi phạm tính phù hợp, nhưng không phù hợp với best practice. Điều kiện MAY là tùy chọn mà nhà phát triển có thể chọn mà không ảnh hưởng tới tính phù hợp.

**Hồ sơ (Profile):** Nói chung là cấu trúc giữ thông tin về Agent hoặc Activity thông qua "tập các tên và tài liệu có ý nghĩa nào đó đối với ứng dụng".

**Đăng ký (Registration):** Một instance của người học đang trải nghiệm một hoạt động cụ thể.

**REST (REpresentational State Transfer):** Kiến trúc dùng để kết nối các dịch vụ web có kết nối mạng. REST dựa trên các phương thức HTTP và tận dụng best practice hiện có của web.

**Dịch vụ:** Thành phần phần mềm chịu trách nhiệm cho một hoặc nhiều khía cạnh của môi trường học tập phân tán. Nói chung, LMS kết hợp nhiều dịch vụ để thiết kế toàn bộ trải nghiệm học tập.

**Statement:** Cấu trúc đơn giản có "kết quả", gồm bộ ba "Actor (người học), Verb, Object" trong ngữ cảnh ghi nhận một khía cạnh của trải nghiệm học tập. Một tập hợp các Statement có thể được dùng để ghi lại thông tin chi tiết đầy đủ về trải nghiệm học tập.

**Tin Can API (TCAPI):** Tên gọi trước đây của API được định nghĩa trong đặc tả này. Được dùng như cách gọi không chính thức của xAPI.

**Verb (động từ):** Trong Statement, Verb định nghĩa hành động mà Actor thực hiện đối với Activity.

### 1.3 4.0 Statement

Giải thích

Statement là khái niệm trung tâm của xAPI. Mọi sự kiện học tập đều được ghi lại dưới dạng Statement. Statement có cấu trúc tương tự câu "I did this".

#### 1.3.1 4.1 Thuộc tính của Statement

**1.3.1.0.1 Chi tiết** Bảng dưới đây trình bày chi tiết từng thuộc tính của Statement.

Tính bất biến của Statement và các ngoại lệ

Statement là bất biến (không thể thay đổi). Dưới đây là các ngoại lệ hoặc các vùng không thuộc phạm vi của quy tắc này.

- Việc gán các thuộc tính có thể hoặc cần thiết trong quá trình xử lý bên trong LRS (`id`, `authority`, `stored`, `timestamp`, `version`).
- Activity được Statement tham chiếu. Nội dung của Activity được tham chiếu trong Statement không được xem là một phần của chính Statement. Điều này có nghĩa là khi Activity được tham chiếu thay đổi, JSON được tạo bởi deep serialization của Statement cũng sẽ thay đổi (xem tham số `format` của Statement API).
- Verb được Statement tham chiếu. Thuộc tính `display` của Verb không được xem là một phần của chính Statement (xem phần từ `format` của Statement API).
- Việc serialization dữ liệu timestamp. Điều này không được xem là một phần của Statement bất biến. Ví dụ: timestamp và các thành phần được ghi trong Statement có thể được trả về theo múi giờ khác với nơi lưu trữ, miễn là thời điểm tham chiếu không bị ảnh hưởng. Chi tiết xem 4.1.7 Timestamp và 4.1.8 Stored.
- Việc serialization các danh sách không có thứ tự. Danh sách Agent trong Group không được coi là danh sách có thứ tự, và LRS có thể trả về danh sách Agent theo thứ tự bất kỳ. Xem 4.1.2.2 Groups.
- Tập đính kèm. Tập đính kèm không phải là một phần của Statement. Ngoài ra, LRS có thể trả về Statement không kèm tập đính kèm khi client yêu cầu. (Chi tiết xem tham số `attachments` trong Statement API.)

#### 1.3.1.0.2 Yêu cầu

- Trong một Statement, không được thiết lập cùng một thuộc tính nhiều lần.
- Statement phải sử dụng `actor`, `verb`, `object`.



- Thứ tự các thuộc tính trong Statement không quan trọng.

Ví dụ Statement tối giản sử dụng toàn bộ thuộc tính được xem là bắt buộc hoặc khuyến nghị:

```
{
  "id": "12345678-1234-5678-1234-567812345678",
  "actor": {
    "mbox": "mailto:xapi@adlnet.gov"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/created",
    "display": {
      "en-US": "created"
    }
  },
  "object": {
    "id": "http://example.adlnet.gov/xapi/example/activity"
  }
}
```

## Appendix A: Ví dụ Statement

### 1.3.1.1 4.1.1 id (định danh) Giải thích

UUID (mọi phiên bản thuộc variant 2 trong RFC 4122) là hợp lệ, và UUID phải ở định dạng chuỗi chuẩn.

Yêu cầu

- Nếu Statement nhận vào không có ID, LRS phải tạo ID.
- ID nên được tạo bởi nhà cung cấp hoạt động.

### 1.3.1.2 4.1.2 Actor (chủ thể) Giải thích

Đối tượng Agent hoặc Group là bắt buộc.

#### 1.3.1.2.1 4.1.2.1 Khi kiểu đối tượng Actor là Agent Giải thích

Agent (cá nhân) là người hoặc hệ thống.

Chi tiết

- Agent phải được định nghĩa bằng một trong bốn loại định danh nghịch hàm (xem 4.1.2.3 Định danh nghịch hàm).
- Agent không được đặt nhiều hơn một định danh nghịch hàm.
- Agent không nên dùng định danh nghịch hàm đang được dùng làm định danh Group.

Bảng thuộc tính của đối tượng Agent:

#### 1.3.1.2.2 4.1.2.2 Khi kiểu đối tượng Actor là Group Giải thích

Group biểu thị tập hợp các Agent và có thể được dùng trong phần lớn tình huống mà Agent có thể được chỉ định. Group có hai loại: ẩn danh và định danh.

Chi tiết

Nhóm ẩn danh dùng để chỉ tập hợp người không có tên gọi cố định, ví dụ đội tạm thời.

Bảng thuộc tính của nhóm ẩn danh:

Nhóm định danh dùng để định danh duy nhất một tập hợp Agent.

Bảng thuộc tính của nhóm định danh:

Yêu cầu

- Hệ thống sử dụng Statement phải coi các nhóm ẩn danh là khác nhau, ngay cả khi chúng có cùng thành viên.
- Khi phát hành nhiều Statement liên quan đến nhóm, tổng hợp dữ liệu, hoặc đăng ký/truy xuất tài liệu, nhà cung cấp hoạt động nên dùng nhóm định danh.
- Nhà cung cấp hoạt động có thể đưa toàn bộ hoặc một phần danh sách Agent vào phần tử `member` của nhóm ẩn danh hoặc nhóm định danh.
- LRS trả về Statement có thể trả về danh sách thành viên trong Group theo bất kỳ thứ tự nào.

Yêu cầu cho nhóm ẩn danh

- Nhóm ẩn danh phải có thuộc tính `member` liệt kê các Agent thành viên.
- Nhóm ẩn danh không được chứa đối tượng Group trong thuộc tính `member`.
- Nhóm ẩn danh không được chứa bất kỳ định danh nghịch hàm nào.

Yêu cầu cho nhóm định danh

- Nhóm định danh phải chứa chính xác một định danh nghịch hàm.
- Nhóm định danh không được chứa đối tượng Group trong thuộc tính `member`.
- Nhóm định danh không nên dùng định danh nghịch hàm cũng đang được dùng làm định danh của Agent.
- Nhóm định danh có thể chứa thuộc tính `member` liệt kê các Agent thành viên.

#### **1.3.1.2.3 4.1.2.3 Định danh nghịch hàm** Giải thích

Định danh nghịch hàm biểu thị giá trị của Agent hoặc nhóm định danh mà việc tham chiếu tới Agent hoặc nhóm định danh đó được đảm bảo về lâu dài trong tương lai.

Bối cảnh

Nếu không thể gắn kết với một cá nhân hoặc nhóm có thể định danh, bản ghi trải nghiệm học tập sẽ không có ý nghĩa. Trong Statement xAPI, điều này được hiện thực bằng tổ hợp các định danh nghịch hàm, tham chiếu nguyên tắc FOAF được chấp nhận rộng rãi (xem Friend Of A Friend).

Chi tiết

Bảng tất cả thuộc tính có thể có của định danh nghịch hàm:

#### **4.1.2.4 Đối tượng Account**

Giải thích

Tài khoản người dùng trong hệ thống sẵn có, như hệ thống không công khai (LMS hoặc intranet) hoặc hệ thống công khai (mạng xã hội).

Chi tiết

- Nếu hệ thống cung cấp đối tượng Account dùng OpenID, nhà cung cấp hoạt động nên dùng thuộc tính OpenID thay vì đối tượng Account.

- Nếu nhà cung cấp hoạt động lo ngại việc công khai thông tin định danh cá nhân của Agent hoặc Group, thì nên dùng tên tài khoản không mang nghĩa (ví dụ: số tài khoản) để vừa bảo toàn ẩn danh vừa định danh được mọi Statement về người đó.

Bảng tất cả thuộc tính của đối tượng Account:

Ví dụ

Ví dụ định danh Agent bằng tên tài khoản không mang nghĩa:

```
{
  "objectType": "Agent",
  "account": {
    "homePage": "http://www.example.com",
    "name": "1625378"
  }
}
```

### 1.3.1.3 4.1.3 Verb (động từ) Giải thích

Verb định nghĩa hành động giữa Actor và Activity.

Bối cảnh

Verb trong Statement xAPI mô tả hành động diễn ra trong trải nghiệm học tập. xAPI không định nghĩa sẵn verb một cách đặc biệt (ngoại lệ là reserved verb <http://adlnet.gov/expapi/verbs/voided>). Thay vào đó, xAPI định nghĩa cách để cộng đồng thực hành tạo verb nhằm giao tiếp giữa các thành viên, đồng thời có thể mở cho dùng chung. Ý tưởng liệt kê trước một danh sách verb cố định có nhiều hạn chế và khó bao phủ hiệu quả mọi trải nghiệm học tập có thể có trong tương lai.

Chi tiết

Verb xuất hiện dưới dạng object trong Statement, gồm IRI và display name thể hiện nghĩa có thể hiểu được bởi con người trên nhiều ngôn ngữ/phương ngữ. Bảng dưới đây liệt kê toàn bộ thuộc tính của đối tượng Verb.

<code>&lt;td&gt;display&lt;/td&gt;</code> <code>&lt;td&gt;&lt;a href="#misclangmap"&gt;Bản đồ ngôn ngữ&lt;/a&gt;&lt;/td&gt;</code> <code>&lt;td&gt;Biểu diễn verb ở dạng con người có thể hiểu được bằng một hoặc nhiều ngôn ngữ.</code> Thuộc tính này không tác động đến ý nghĩa của Statement, nhưng cung cấp cách biểu <code>&lt;td&gt;Khuyến nghị&lt;/td&gt;</code>
--

Yêu cầu cho Verb ID

- Hệ thống diễn giải Statement phải dùng Verb IRI để xác định ý nghĩa.
- IRI trong ID nên bao gồm phần con người có thể đọc được, cung cấp đủ thông tin nghĩa để phân biệt verb mục tiêu với các verb tương tự về ngữ pháp.
- Một Verb IRI không được dùng để biểu thị nhiều nghĩa.

Yêu cầu cho hiển thị Verb ở AP

- Thuộc tính `display` nên được dùng trong mọi Statement.
- Thuộc tính `display` phải được dùng để biểu thị nghĩa đã được định nghĩa bởi Verb IRI.

Yêu cầu cho hiển thị Verb ở LRS

Các yêu cầu dưới đây liên quan đến thuộc tính `display` do LRS trả về qua API.

- Nếu Statement được gọi với `format` là `exact`, LRS phải trả về thuộc tính `display` đúng như trong Statement (hoặc đúng trạng thái bị lược bỏ).
- Nếu Statement được gọi với `format` là `ids`, LRS không nên bao gồm thuộc tính `display`.
- Nếu Statement được gọi với `format` là `canonical`, LRS nên trả về `display` đã chuẩn hóa của verb.
- LRS có thể xác định `display` chuẩn hóa dựa trên thuộc tính `display` của verb trong Statement nhận được, thuộc tính `name` trong metadata nêu tại section 5.4 Identifier metadata, hoặc `display` verb được định nghĩa ở nơi khác.

Yêu cầu cho client hiển thị Verb

Các yêu cầu dưới đây liên quan đến thuộc tính `display` khi hiển thị cho người dùng bởi LRS hoặc hệ thống khác.

- Không được dùng thuộc tính `display` để bóp méo ý nghĩa của verb.
- Hệ thống nhận Statement không được dùng thuộc tính `display` để suy diễn bất kỳ ý nghĩa nào của Statement.
- Hệ thống nhận Statement không được dùng thuộc tính `display` cho bất kỳ mục đích nào ngoài hiển thị cho con người.

Việc dùng thuộc tính `display` để tổng hợp hoặc phân loại Statement là vi phạm các yêu cầu trên.

- Hệ thống hiển thị verb của Statement trên giao diện người dùng có thể chọn hiển thị một trong các nguồn sau: thuộc tính `display` trong Statement, thuộc tính `name` trong metadata nêu tại section 5.4 Identifier metadata, hoặc `display` verb được định nghĩa ở nơi khác.
- Hệ thống hiển thị verb của Statement không được hiển thị từ có nghĩa khác với nghĩa của verb, nhưng có thể thay đổi cách diễn đạt hoặc thì để dễ đọc hơn.

Ví dụ

Ví dụ verb có chứa thuộc tính khuyến nghị:

```
{
  "id": "http://www.adlnet.gov/XAPIprofile/ran(travelled_a_distance)",
  "display": {
    "en-US": "ran",
    "es" : "corrió"
  }
}
```

Verb trong ví dụ trên chỉ nhằm mục đích minh họa. Điều này không có nghĩa verb mang nghĩa đó đã được định nghĩa bằng ID này. Nguyên tắc này áp dụng cho mọi ví dụ verb trong đặc tả này, ngoại trừ reserved verb (<http://adlnet.gov/expapi/verbs/voided>).

#### 1.3.1.3.1 4.1.3.1 Cách dùng về ngôn ngữ và ý nghĩa của Verb Chi tiết

*Ý nghĩa*

IRI biểu diễn bởi Verb ID biểu thị một nghĩa cụ thể của từ, không phải chỉ bản thân từ đó.

Ví dụ, từ tiếng Anh `fired` có thể mang nhiều nghĩa theo ngữ cảnh như ”nổ súng”, ”nung trong lò”, hoặc ”sa thải nhân viên”. Trong ví dụ này, IRI biểu thị một nghĩa cụ thể trong số đó.

Thuộc tính `display` cho phép mức linh hoạt nhất định về thì. Verb IRI thường được kỳ vọng ở thì quá khứ, nhưng với Activity cụ thể, nếu dùng thì khác (với cùng verb) hợp lý hơn thì có thể dùng như vậy.

## Ngôn ngữ

Verb trong xAPI là IRI và biểu thị ý nghĩa riêng không phụ thuộc vào ngôn ngữ cụ thể nào.

Ví dụ, Verb IRI như `http://example.org/firearms#fire` biểu thị hành động nổ súng; còn Verb IRI như `http://example.com/انددنن/خوجلل` biểu thị hành động đọc sách.

### 1.3.1.3.2 4.1.3.2 Cách dùng theo cộng đồng thực hành Giải thích

Cộng đồng thực hành, để đáp ứng yêu cầu của thành viên, sẽ đến lúc cần định nghĩa verb mới.

Vì vậy, cộng đồng thực hành xAPI xây dựng hồ sơ cộng đồng dựa trên thuật ngữ kiểm soát mới hoặc hiện có.

Yêu cầu cho cộng đồng thực hành

- Khi định nghĩa verb mới, cần sở hữu IRI hoặc có sự cho phép từ chủ sở hữu để dùng IRI đó cho verb xAPI.
- Khi định nghĩa verb mới, nên cung cấp định nghĩa con người có thể hiểu được về cách dùng dự kiến của verb và để định nghĩa đó truy cập được qua IRI.

Yêu cầu cho nhà cung cấp hoạt động

- Nhà cung cấp hoạt động nên dùng verb tương ứng đã tồn tại khi có thể.
- Nếu không có verb phù hợp, nhà cung cấp hoạt động có thể tạo và dùng verb mới.

### 1.3.1.4 4.1.4 Object (đối tượng) Giải thích

Object trong Statement có thể là Activity, Agent/Group, Sub-Statement, hoặc tham chiếu tới Statement. Object là phần biểu diễn "đối tượng" trong Statement. Ví dụ, trong Statement "I did this", Object tương ứng phần "this".

Ví dụ:

- Khi object là Activity: "Jeff đã viết một bài luận về leo núi."
- Khi object là Agent: "Nellie đã phỏng vấn Jeff."
- Khi object là Sub-Statement hoặc Statement Reference (khác nhau về cách biểu diễn nhưng con người đều hiểu được): "Nellie đã bình luận về việc Jeff đã viết một bài luận về leo núi."

Chi tiết

Object được cung cấp trong thuộc tính này nên có thuộc tính `objectType`. Nếu không chỉ định, `objectType` được hiểu là `Activity`. Các giá trị hợp lệ khác gồm `Agent`, `Group`, `Sub-Statement`, hoặc `StatementRef`. Thuộc tính của object thay đổi theo `objectType`.

#### 1.3.1.4.1 4.1.4.1 Khi ObjectType là Activity Chi tiết

Statement có thể chỉ ra Activity làm object của Statement. Thuộc tính object trong trường hợp này như bảng dưới.

Nếu cùng một ID có thể dùng cho hai Activity khác nhau, thì tính hợp lệ của các Activity này sẽ bị nghi ngờ. Điều đó có nghĩa là LRS, dù có chủ ý, cũng không thể xử lý cùng một Activity ID như tham chiếu tới hai Activity khác nhau. Nói cách khác, khi xung đột với hệ thống khác xảy ra, không thể suy ra đúng chủ ý.

Định nghĩa Activity

Đối tượng định nghĩa Activity có các thuộc tính như bảng dưới.

**Ghi chú:** IRI fragment (còn gọi là IRI tương đối) không phải IRI hợp lệ. Cũng như Verb, nhà cung cấp hoạt động được khuyến nghị tìm và dùng loại Activity đã được thiết lập, áp dụng rộng rãi.

### Yêu cầu cho Activity ID

- Activity ID phải là duy nhất.
- Activity ID phải luôn tham chiếu cùng một Activity.
- Activity ID nên dùng miền mà người tạo được cho phép sử dụng cho mục đích này.
- Activity ID nên được tạo theo lược đồ bảo đảm mọi Activity ID trong miền đó là duy nhất.
- Activity ID có thể trỏ tới metadata hoặc IRL của Activity.

### Yêu cầu cho LRS

- Nếu một Activity ID được xem là đang được dùng bởi nhiều tác giả hoặc tổ chức, LRS không được thực hiện hành động.
- LRS không được xử lý nhiều tham chiếu đến cùng một ID thành nhiều tham chiếu đến các Activity khác nhau.
- Nếu nhận Statement có định nghĩa Activity khác với định nghĩa đã lưu, LRS nên xác định liệu nhà cung cấp hoạt động có quyền thay đổi định nghĩa hay không; nếu có, nên cập nhật định nghĩa Activity đã ghi.
- LRS có thể chấp nhận chỉnh sửa nhỏ trong định nghĩa Activity, ví dụ sửa chính tả. Tuy nhiên không được chấp nhận thay đổi làm đổi đáp án đúng.

### Yêu cầu cho nhà cung cấp hoạt động

- Nhà cung cấp hoạt động phải bảo đảm Activity ID không bị dùng trùng cho nhiều Activity.
- Nhà cung cấp hoạt động chỉ được tạo state hoặc Statement cho Activity ID tương thích và nhất quán với state/Statement đã ghi trước đó cho cùng ID.
- Nhà cung cấp hoạt động không được phá vỡ tính tương thích khi cập nhật phiên bản Activity (theo revision hoặc nền tảng khác).

### Yêu cầu cho metadata

- Nếu IRI của Activity là IRL, nên thử GET IRL đó với HTTP header `Accept: application/json, */*`. LRS phải thực hiện ngay khi phát hiện Activity ID.
- Khi tải JSON định nghĩa Activity hợp lệ từ IRL dùng làm Activity ID, LRS nên hợp nhất định nghĩa đã tải vào định nghĩa nội bộ của Activity, đồng thời giữ lại tên/định nghĩa không có trong phần mới tải.
- Activity có định danh IRL có thể cung cấp metadata theo định dạng JSON của Activity Definition dùng `Content-Type: application/json`.
- Khi xác định biểu diễn nội bộ của định nghĩa Activity, LRS có thể phân tích định nghĩa từ IRL dùng làm Activity ID; và khi đọc tài liệu tùy ý từ đó, LRS có thể cân nhắc định nghĩa này.

#### 1.3.1.4.2 Interaction Activities Bối cảnh

E-learning truyền thống có cấu trúc tích hợp Interaction và Assessment. Để mở rộng thực tiễn và cấu trúc đó sang xAPI, đặc tả này bao gồm định nghĩa Interaction tham chiếu mô hình dữ liệu SCORM 2004 4th Edition. Các định nghĩa này nhằm cung cấp cơ chế đơn giản và quen thuộc để ghi lại dữ liệu Interaction. Từ phiên bản 1.0.3, các tham chiếu trực tiếp đến mô hình dữ liệu SCORM bắt đầu được loại bỏ dần, và các yêu cầu liên quan được mô tả trực tiếp trong tài liệu này.

Các định nghĩa Interaction này đơn giản, dễ dùng nhưng cũng có giới hạn. Cộng đồng thực hành cần định nghĩa Interaction nâng cao có thể thực hiện bằng cách dùng mở rộng của loại Activity và định nghĩa Activity.

#### Chi tiết

Bảng dưới mô tả các thuộc tính của Interaction Activity.

Loại interaction

Bảng dưới mô tả loại interaction được biểu thị bởi mỗi `interactionType`. Các loại interaction này dựa trên các loại cho phép trong `cmi.interactions.n.type` của SCORM 2004 4th Edition Runtime Environment. Xem Appendix C cho từng loại interaction.

#### Mẫu trả lời

Bảng dưới mô tả định dạng chuỗi của phần tử `correctResponsesPattern` cho từng loại interaction. Định dạng này cũng được dùng để biểu diễn câu trả lời của người học trong object yêu cầu. Các loại interaction này dựa trên `cmi.interactions.n.type` của SCORM 2004 4th Edition Runtime Environment. Xem Appendix C cho ví dụ từng định dạng.

#### Tham số chuỗi

Một số giá trị trong mẫu trả lời ở trên có thể gắn thêm tham số. Chúng dựa trên chuỗi phân cách do SCORM 2004 4th Edition Runtime Environment định nghĩa. Các tham số được biểu diễn theo dạng `{parameter=value}`. Xem ví dụ long-fill-in trong Appendix C.

Các tham số dưới đây hợp lệ ở đầu chuỗi định nghĩa danh sách phần tử cho các loại interaction dạng danh sách:

Tham số dưới đây hợp lệ ở đầu danh sách phần tử của các loại interaction dạng danh sách:

#### Yêu cầu

- Interaction Activity phải chứa `interactionType` hợp lệ.
- Interaction Activity nên có loại Activity là `http://adlnet.gov/expapi/activities/cmi.interaction`.
- Khi nhận `interactionType` hợp lệ, LRS có thể kiểm tra các thuộc tính còn lại theo bảng dưới; nếu không hợp lệ với Interaction Activity thì có thể trả HTTP 400 Bad Request.
- LRS không nên đặt giới hạn độ dài ký tự cho mẫu trả lời.
- LRS không nên đặt giới hạn độ dài mảng `correctResponsesPatterns` cho bất kỳ `interactionType` nào.

#### 1.3.1.4.3 Thành phần interaction Chi tiết

Định nghĩa thành phần interaction như sau:

Tùy theo `interactionType`, Interaction Activity có thể có thêm thuộc tính chứa các phần tử interaction tương ứng. Các thuộc tính này gọi là "interaction component lists". Bảng sau mô tả các danh sách thành phần interaction được hỗ trợ cho từng `interactionType`.

#### Yêu cầu

- Trong mảng thành phần interaction, mọi ID phải khác nhau.
- ID của thành phần interaction không được chứa khoảng trắng.

#### Ví dụ

Định nghĩa Activity cho từng loại `cmi.interaction` xem mẫu tại Appendix C.

#### 1.3.1.4.4 4.1.4.2 Khi object là một Agent hoặc Group đơn lẻ Yêu cầu

- Statement tham chiếu Agent hoặc Group đơn lẻ bắt buộc phải khai báo rõ thuộc tính `objectType`.

Chi tiết về Agent xem Section 4.1.2 Actor.

#### 1.3.1.4.5 4.1.4.3 Khi object là Statement Bối cảnh

Có hai trường hợp object là Statement. Thứ nhất, object có thể ở dạng tham chiếu tới Statement đã tồn tại. Ví dụ phổ biến là gắn đánh giá hoặc bình luận cho một trải nghiệm có thể được xem là một sự kiện độc lập. Thứ hai, object có thể ở dạng Statement độc lập thông qua Sub-Statement. Cách dùng phổ biến của Sub-Statement là biểu diễn trải nghiệm mà nếu viết thành Statement đơn lẻ có thể gây hiểu nhầm.

#### 1.3.1.4.6 Statement Reference Giải thích

Statement Reference là con trỏ tới một Statement đã tồn tại khác.

Yêu cầu

- Statement Reference phải đặt objectType là StatementRef.
- Statement Reference phải đặt UUID của Statement vào thuộc tính id. LRS không có nghĩa vụ xác minh UUID đó trùng với Statement có thật.

Bảng thuộc tính của đối tượng Statement Reference:

Ví dụ

Giả sử Statement có ID 8f87ccde-bb56-4c2e-ab83-44982ef22df0 đã được ghi, ví dụ dưới cho thấy cách phát hành bình luận tới Statement gốc bằng một Statement mới.

```
{
  "actor" : {
    "objectType": "Agent",
    "mbox": "mailto:test@example.com"
  },
  "verb" : {
    "id": "http://example.com/commented",
    "display": {
      "en-US": "commented"
    }
  },
  "object" : {
    "objectType": "StatementRef",
    "id": "8f87ccde-bb56-4c2e-ab83-44982ef22df0"
  },
  "result" : {
    "response" : "Wow, nice work!"
  }
}
```

#### 1.3.1.4.7 Sub-Statement Giải thích

Sub-Statement là cấu trúc giống Statement nhưng là một phần của Statement, và (khác Statement) không tương ứng trực tiếp với sự kiện thực tế đã xảy ra. Ví dụ, có thể dùng để mô tả mẫu Statement có thể xảy ra trong tương lai, như hành vi giáo viên yêu cầu khi đánh giá học viên (mà chưa cần học viên thực sự thực hiện hành vi đó).

Yêu cầu

- Sub-Statement phải khai báo rõ objectType là SubStatement.
- Sub-Statement phải được đánh giá như Statement, ngoài các yêu cầu riêng của Sub-Statement.



- Sub-Statement không được có thuộc tính id, stored, version, hoặc authority.
- Sub-Statement không được chứa Sub-Statement khác (không được lồng nhau).

Ví dụ

Một cách dùng đáng chú ý của Sub-Statement là tạo Statement biểu thị ý định. Ví dụ, có thể dùng Sub-Statement để tạo dạng "<I> <planned> (<I> <did> <this>)" nhằm chỉ việc dự định thực hiện một hành động. Ví dụ cụ thể bên dưới biểu thị "I planned to visit 'Some Awesome Website'".

```
{
  "actor": {
    "objectType": "Agent",
    "mbox": "mailto:test@example.com"
  },
  "verb": {
    "id": "http://example.com/planned",
    "display": {
      "en-US": "planned"
    }
  },
  "object": {
    "objectType": "SubStatement",
    "actor": {
      "objectType": "Agent",
      "mbox": "mailto:test@example.com"
    },
    "verb": {
      "id": "http://example.com/visited",
      "display": {
        "en-US": "will visit"
      }
    },
    "object": {
      "objectType": "Activity",
      "id": "http://example.com/website",
      "definition": {
        "name": {
          "en-US": "Some Awesome Website"
        }
      }
    }
  }
}
```

#### 1.3.1.5 4.1.5 Result (kết quả) Giải thích

Thuộc tính tùy chọn để đo kết quả xử lý của Statement.

Chi tiết

Bảng dưới liệt kê các thuộc tính của đối tượng Result.

## Yêu cầu

- Thuộc tính `duration` phải biểu diễn bằng khoảng thời gian theo định dạng của ISO 8601:2004(E) section 4.4.3.2. Không được dùng định dạng xấp xỉ (ISO 8601:2004(E) section 4.4.3.3).

### 1.3.1.5.1 4.1.5.1 Score (điểm) Giải thích

Thuộc tính số tùy chọn biểu thị kết quả Activity đã được phân loại mà Agent đạt được.

#### Chi tiết

Bảng dưới định nghĩa đối tượng Score.

Các thuộc tính của đối tượng Score dựa trên các thuộc tính tương ứng của `cmi.score` được định nghĩa trong SCORM 2004 4th Edition.

## Yêu cầu

- Nếu biết điểm theo tỷ lệ phần trăm hợp lý, đối tượng Score nên bao gồm `scaled`.
- Không nên dùng đối tượng Score cho điểm liên quan đến tiến độ hoặc hoàn thành. Thay vào đó, cân nhắc dùng Extensions.

### 1.3.1.6 4.1.6 Context (ngữ cảnh) Giải thích

Thuộc tính tùy chọn để gắn thông tin phụ thuộc ngữ cảnh vào Statement. Tất cả thuộc tính đều là tùy chọn.

#### Bối cảnh

Thuộc tính `context` tạo cơ hội thêm thông tin ngữ cảnh vào Statement. Ví dụ, khi trải nghiệm diễn ra như một phần của hoạt động nhóm, có thể ghi tên người hướng dẫn của trải nghiệm đó, hoặc ghi cách trải nghiệm được gắn vào một hoạt động lớn hơn.

#### Chi tiết

## Yêu cầu

- Khi dùng thuộc tính `revision`, object của Statement chỉ được là Activity.
- Khi dùng thuộc tính `platform`, object của Statement chỉ được là Activity.
- Nếu không áp dụng được hoặc không biết, không được dùng thuộc tính `language`.
- Thuộc tính `revision` nên dùng để ghi các sửa lỗi nhỏ (ví dụ lỗi chính tả).
- Không nên dùng thuộc tính `revision` khi có thay đổi lớn ở mục tiêu học tập, phương pháp giảng dạy hoặc học liệu của Activity. Trong trường hợp đó nên dùng Activity mới.

*Ghi chú:* Revision không ảnh hưởng đến hành vi trong phạm vi xAPI. Nó chỉ được ghi lại để công cụ báo cáo có thể sử dụng.

### 1.3.1.6.1 4.1.6.1 Thuộc tính Registration Giải thích

Instance của người học đang thực hiện một hoạt động học tập cụ thể.

#### Chi tiết

Nếu LRS là phần không thể tách rời của LMS, LMS được kỳ vọng có khái niệm registration. Trong xAPI, khái niệm registration được áp dụng rộng hơn. Registration có thể được xem liên quan đến lần thử học hoặc giai đoạn học, và một bản ghi có thể trải trên các hoạt động thuộc registration. Việc hoàn thành hoạt động học tập cũng không nhất thiết đồng nghĩa kết thúc registration. Registration cũng không giới hạn ở một Agent duy nhất.

### 1.3.1.6.2 4.1.6.2 Thuộc tính ContextActivities Giải thích

Map các loại context của hoạt động học tập mà Statement này liên quan tới.

Bối cảnh

Nhiều Statement không chỉ chứa một object Activity mục tiêu hiện tại mà còn liên quan đến các hoạt động khác trong ngữ cảnh. "Context activities" cho phép biểu diễn có cấu trúc các hoạt động liên quan này.

Chi tiết

Có 4 loại context hợp lệ. Trong Statement đích, có thể dùng tất cả, dùng một phần, hoặc không dùng loại nào.

1. **Parent:** Activity có liên hệ trực tiếp với Activity là object của Statement. Trong đa số trường hợp sẽ có một parent rõ ràng hoặc không có parent, hiếm khi có nhiều parent. Ví dụ: Statement về một câu hỏi trong bài kiểm tra sẽ có parent Activity là "bài kiểm tra".
2. **Grouping:** Activity có liên hệ gián tiếp với Activity là object của Statement. Ví dụ: một khóa học thuộc một chứng chỉ. Khóa học có quan hệ parent với từng buổi học, còn chứng chỉ có quan hệ grouping với các buổi học.
3. **Category:** Activity dùng để phân loại Statement, đồng nghĩa với "tag". Category không chỉ dùng cho phân loại mà còn nên dùng để chỉ ra community profile liên quan hành vi xAPI.

Ví dụ: Anna làm bài kiểm tra sinh học, và Statement được ghi bằng profile CMI-5. Activity của Statement tham chiếu bài kiểm tra, còn category là profile CMI-5.

4. **Other:** Context Activity không phù hợp với các thuộc tính còn lại. Ví dụ: Anna học bằng sách giáo khoa cho bài kiểm tra sinh học. Activity của Statement tham chiếu sách giáo khoa, còn bài kiểm tra là context activity loại other.

Để Statement phiên bản 0.95 tương thích với phiên bản 1.0.0, có thể dùng một Activity Object đơn lẻ làm giá trị.

**Ghi chú:** Phần này không nhằm mô tả toàn bộ quan hệ có thể có của Statement object. Nó mô tả quan hệ phù hợp với Statement cụ thể (dù bản chất object thường quan trọng khi xác định). Ví dụ, với Statement về một bài kiểm tra, đưa khóa học chứa bài kiểm tra đó vào parent là hợp lý; nhưng liệt kê toàn bộ chương trình học vì có thể liên quan vào grouping thì không hợp lý.

Yêu cầu

- Mọi key của đối tượng contextActivities phải là một trong parent, grouping, category, hoặc other.
- Mọi value của đối tượng contextActivities phải là một Activity Object hoặc mảng Activity Object.
- LRS phải trả các value của contextActivities dưới dạng mảng, ngay cả khi nhận vào là một Activity Object đơn.
- Khi trả về một Activity Object đơn, LRS phải trả mảng có một phần tử chứa Activity đó.
- Client nên đảm bảo mọi value trong contextActivities là mảng Activity Object, không ở dạng Activity Object đơn.

Ví dụ

Xét cấu trúc phân cấp sau: "Câu hỏi 1 đến 6" thuộc "Bài kiểm tra 1", và "Bài kiểm tra 1" thuộc khóa "Đại số 1". Sáu câu hỏi được ghi là một phần của bài kiểm tra bằng cách khai báo "Bài kiểm tra 1" là parent. Chúng cũng được grouping cùng các Statement khác của "Đại số 1" để sao chép đầy đủ phân cấp cấp cao. Điều này đặc biệt hữu ích khi object của Statement là Agent (không phải Activity), ví dụ: "Andrew đã hướng dẫn Ben trong context của Đại số 1".

```
{
  "parent" : [
    {"id" : "http://example.adlnet.gov/xapi/example/test1"}
  ],
  "grouping" : [
    {"id" : "http://example.adlnet.gov/xapi/example/Algebra1"}
  ]
}
```

#### 1.3.1.7 4.1.7 Timestamp (mốc thời gian) Giải thích

Thời điểm trải nghiệm xảy ra.

Chi tiết

Timestamp được định dạng theo chuẩn thông thường của ISO 8601 và tương ứng với thời điểm sự kiện mô tả trong Statement xảy ra. Nếu không có trong Statement khi gửi tới LRS, LRS sẽ điền cùng giá trị như khi đặt cho Stored.

Timestamp trong Statement có thể khác Stored (thời điểm Statement được lưu). Ví dụ, có thể có độ trễ giữa lúc trải nghiệm xảy ra và lúc LRS nhận Statement. Nếu trải nghiệm xảy ra trong một khoảng thời gian, timestamp có thể biểu thị thời điểm bắt đầu, kết thúc, hoặc bất kỳ thời điểm nào trong khoảng trải nghiệm. Cộng đồng thực hành được kỳ vọng xác định thời điểm phù hợp để ghi timestamp cho từng loại trải nghiệm. Ví dụ, với trải nghiệm ăn tại nhà hàng có thể phù hợp ghi timestamp bắt đầu; trong khi với trải nghiệm hoàn thành chứng chỉ có thể phù hợp ghi timestamp kết thúc. Các ví dụ này chỉ có tính minh họa, không mang tính quy phạm.

Yêu cầu

- timestamp phải theo định dạng ISO 8601.
- Nếu thuộc tính timestamp không được chỉ định, LRS nên đặt bằng giá trị của Stored.
- timestamp nên bao gồm múi giờ.
- Nếu timestamp có múi giờ, LRS có thể trả về bằng múi giờ khác miễn là thời điểm tham chiếu không đổi. LRS nên trả về múi giờ UTC.
- timestamp có thể chỉ định bất kỳ điểm nào trong thời lượng trải nghiệm, không chỉ điểm đầu hay cuối.
- timestamp có thể bị cắt cụt hoặc làm tròn đến độ chính xác tối thiểu 3 chữ số thập phân giây (phải giữ độ chính xác mili giây).
- AP không được chỉ định thời điểm tương lai làm timestamp trong Statement.
- LRS phải từ chối timestamp tương lai để tránh vấn đề do sai lệch đồng hồ.
- Sub-Statement có thể có timestamp chỉ thời điểm tương lai.

#### 1.3.1.8 4.1.8 Stored Giải thích

Thời điểm Statement được ghi vào LRS.

Chi tiết

Thuộc tính stored là thời điểm chính xác Statement được ghi. LRS dùng Timestamp để ghi thời điểm trải nghiệm mô tả trong Statement xảy ra.

Yêu cầu

- Thuộc tính stored phải theo định dạng ISO 8601 ([https://en.wikipedia.org/wiki/ISO\\_8601#Combined\\_date\\_and\\_time\\_representations](https://en.wikipedia.org/wiki/ISO_8601#Combined_date_and_time_representations)).

- Thuộc tính `stored` phải do LRS thiết lập. LRS sẽ xác minh và ghi đè giá trị `stored` trong Statement nhận vào.
- Thuộc tính `stored` nên bao gồm múi giờ.
- Nếu `stored` có múi giờ, LRS có thể trả về bằng múi giờ khác miễn là thời điểm tham chiếu không đổi. LRS nên trả về múi giờ UTC.
- Thuộc tính `stored` nên là thời điểm hiện tại hoặc quá khứ.
- Thuộc tính `stored` có thể bị cắt cụt hoặc làm tròn đến độ chính xác tối thiểu 3 chữ số thập phân giây (phải giữ độ chính xác mili giây).

#### 1.3.1.9 4.1.9 Authority (thẩm quyền) Giải thích

Thuộc tính `authority` cung cấp thông tin "ai hoặc cái gì khẳng định Statement này là đúng".

Chi tiết

Khẳng định thẩm quyền nghĩa là xác thực người dùng trên hệ thống hoặc ứng dụng.

Yêu cầu

- Authority, trừ trường hợp 3-legged OAuth, phải là một Agent trong các tình huống nếu không sẽ hình thành Group gồm 2 Agent (ứng dụng và người dùng).
- Khi người dùng kết nối trực tiếp bằng HTTP Basic Auth hoặc là thành viên của Group, LRS phải đưa người dùng đó vào như Agent có toàn quyền.
- LRS phải xác nhận mọi Statement được ghi đều có authority.
- LRS nên ghi đè authority trong mọi Statement nhận vào dựa trên thông tin xác thực dùng để gửi Statement.
- LRS có thể giữ nguyên authority nhận vào, nhưng chỉ nên khi đã có quan hệ tin cậy mạnh, và cần cực kỳ thận trọng.
- Khi người dùng kết nối trực tiếp bằng HTTP Basic Auth hoặc là một phần của 3-legged OAuth, LRS có thể định danh người dùng bằng thuộc tính định danh hợp lệ.

##### 1.3.1.9.1 Thông tin xác thực OAuth dưới dạng authority Giải thích

Đây là luồng sử dụng OAuth. Hỗ trợ cả 2-legged và 3-legged OAuth.

Chi tiết

Quy trình này giả định Statement được lưu qua kết nối OAuth đã xác minh, và LRS tạo hoặc thay đổi thuộc tính `authority` của Statement.

Trong workflow 3-legged OAuth, xác thực bao gồm cả "OAuth consumer" và "người dùng của OAuth service provider". Ví dụ, yêu cầu từ plugin Twitter đã được cấp quyền trên tài khoản Facebook chứa thông tin xác thực không chỉ cho ứng dụng khách (Twitter) hoặc người dùng, mà còn cho mối liên kết duy nhất giữa hai bên.

Yêu cầu

- `authority` phải chứa Agent Object biểu diễn OAuth consumer, hoặc chính Agent Object đó, hoặc (trong 3-legged OAuth) Agent Object như một phần của Group.
- Agent biểu diễn OAuth consumer phải được định danh bằng `account`.
- Agent biểu diễn OAuth consumer phải dùng `consumer key` làm giá trị thuộc tính `account name`.
- Nếu Agent biểu diễn OAuth consumer là ứng dụng đã đăng ký, `endpoint token request` phải được dùng làm `homepage` của `account`.
- Nếu Agent biểu diễn OAuth consumer không phải ứng dụng đã đăng ký, `endpoint temporary credential request` phải được dùng làm `homepage` của `account`.

- LRS không được tin cậy phần ứng dụng trong authority từ cùng nguồn với ứng dụng chưa đăng ký có cùng account name. (Nhiều ứng dụng chưa đăng ký có thể chọn cùng consumer key; do đó không có cách tin cậy để xác minh cặp temporary credential và account name.)
- Mỗi consumer chưa đăng ký nên dùng consumer key duy nhất.

Ví dụ

Cặp OAuth consumer và người dùng:

```
?{"authority": {
  "objectType" : "Group",
  "member": [
    {
      "account": {
        "homePage": "http://example.com/xAPI/OAuth/Token",
        "name": "oauth_consumer_x75db"
      }
    }, {
      "mbox": "mailto:bob@example.com"
    }
  ]
}
```

#### 1.3.1.10 4.1.10 Version (phiên bản) Giải thích

Hệ thống xử lý dữ liệu từ LRS có thể quyết định hành vi dựa trên thông tin phiên bản của Statement. Vì mô hình dữ liệu Statement được đảm bảo nhất quán xuyên suốt mọi phiên bản 1.0.x, LRS được cho phép linh hoạt hơn trong phiên bản Statement có thể chấp nhận để hỗ trợ luồng dữ liệu giữa các LRS.

Yêu cầu

- Version phải tuân theo layout của API version header trong đặc tả API Versioning.

Yêu cầu cho LRS

- Với Statement hợp lệ, LRS phải chấp nhận mọi Statement có version bắt đầu bằng 1.0..
- LRS phải từ chối mọi Statement có version không bắt đầu bằng 1.0..
- Statement do LRS trả về phải giữ phiên bản tại thời điểm được chấp nhận. Nếu không có thông tin phiên bản, version phải đặt là 1.0.0.

Yêu cầu cho client

- Nếu client đặt version cho Statement thì phải là 1.0.0.
- Client không nên đặt version cho Statement.

#### 1.3.1.11 4.1.11 Attachments (tệp đính kèm) Giải thích

Tài liệu số cung cấp bằng chứng cho trải nghiệm học tập.

Bối cảnh

Trong một số trường hợp, tài liệu đính kèm là phần quan trọng về mặt logic của hồ sơ học tập. Hãy nghĩ đến mô phỏng liên lạc với kiểm soát không lưu, bài luận, video. Ví dụ khác là ảnh chứng chỉ hoàn thành được cấp sau trải nghiệm. Có khả năng ghi các tài liệu đính kèm này vào LRS và đọc lại từ LRS là rất hữu ích. Nếu muốn đính tài liệu vào Sub-Statement, khuyến nghị mạnh rằng hãy đưa nó vào Statement thường như một payload dưới dạng Statement Attachment Object.

## Chi tiết

Bảng dưới mô tả toàn bộ thuộc tính của đối tượng Attachment.

### *Quy trình trao đổi Attachment*

1. Statement chứa Attachment được cấu trúc theo định dạng truyền tải mô tả bên dưới.
2. Statement được gửi tới hệ thống nhận với content type `multipart/mixed`, Attachment được đặt ở cuối bản truyền.
3. Hệ thống nhận quyết định chấp nhận hay từ chối Statement dựa trên thông tin phần đầu.
4. Khi chấp nhận Attachment, có thể đối chiếu header Attachment trong Statement với dữ liệu thô bằng cách so sánh SHA-2 của dữ liệu thô với SHA-2 khai báo ở header. Không được đối chiếu bằng phương pháp khác.

### Yêu cầu cho Statement batch có Attachment

Statement batch có Attachment, Statement result, hoặc Statement đơn phải thỏa một trong các điều kiện sau:

- Với `application/json`, phải chứa `fileUrl` cho mọi Attachment (trừ kết quả Statement khi attachment filter là false).
- Hoặc tuân theo định nghĩa `multipart/mixed` của RFC1341 ([http://www.w3.org/Protocols/rfc1341/7\\_2\\_Multipart.html](http://www.w3.org/Protocols/rfc1341/7_2_Multipart.html)), đồng thời thỏa các điều kiện:
  - Phần đầu của tài liệu multipart chứa chính Statement với type `application/json`.
  - Các phần còn lại chứa dữ liệu thô Attachment, là phần logic của Statement. Cơ chế này khả dụng khi phát PUT/POST lên Statement resource.
  - Header của mỗi phần sau phần đầu (Statement) phải chứa tham số `X-Experience-API-Hash`.
  - Header của mỗi phần sau phần đầu (Statement) phải chứa `Content-Transfer-Encoding: binary`.
  - Khi gửi đồng thời nhiều Statement dùng chung một Attachment, chỉ nên chứa một bản dữ liệu Attachment đó.
  - Header mỗi phần nên có tham số `Content-Type`; riêng phần đầu (chứa Statement) bắt buộc là `application/json`.
  - Nếu một tham số có thuộc tính tương ứng trong Attachment Object, và cả hai cùng được chỉ định cho cùng Attachment, thì giá trị phải nhất quán.

### Yêu cầu cho LRS

- Khi client yêu cầu, LRS phải bao gồm Attachment theo định dạng truyền tải nêu trên (xem 7.2 Statement API).
- LRS không được kéo Statement từ LRS khác nếu không yêu cầu Attachment.
- Nếu có dữ liệu Attachment đã nhận, LRS không được gửi Statement đó tới LRS khác mà bỏ qua dữ liệu đính kèm.
- Khi nhận tài liệu type `application/json` qua PUT/POST, LRS phải chấp nhận Statement batch không chứa Attachment Object.
- Khi nhận tài liệu type `application/json` qua PUT/POST, LRS phải chấp nhận Statement batch chỉ chứa Attachment Object có `fileUrl` đã cung cấp.
- Khi nhận tài liệu type `multipart/mixed` qua PUT/POST, LRS phải chấp nhận Statement batch có Attachment theo định dạng truyền tải ở trên.
- Khi nhận tài liệu type `multipart/mixed` qua PUT/POST, LRS phải từ chối Statement batch có Attachment nếu không có `fileUrl`, hoặc nếu một phần Attachment nhận vào không khớp hash.
- Khi nhận tài liệu type `multipart/mixed` qua PUT/POST, LRS nên giả định `Content-Transfer-Encoding` của phần Attachment là nhị phân.
- LRS có thể từ chối Statement (batch) có kích thước vượt giới hạn cấu hình cho phép.

**Ghi chú:** Nếu Statement batch dùng định dạng mime/multipart có chứa Attachment thì không có yêu cầu bổ sung khác.

Yêu cầu cho client

- Client có thể gửi Statement chứa Attachment theo mô tả ở trên.
- Khi dùng POST, client có thể gửi nhiều Statement, trong đó một phần hoặc toàn bộ có Attachment.
- Client có thể bỏ qua toàn bộ yêu cầu của định dạng multipart/mixed và gửi batch application/json khi mọi Attachment Object đều có `fileUrl`.

Ví dụ

Đây là ví dụ rất đơn giản của Statement có Attachment. Lưu ý:

- Boundary trong ví dụ được chọn để minh họa tập ký tự hợp lệ.
- Boundary được chọn không khớp với bất kỳ phần nào của từng Attachment đã mã hóa.
- Để dễ đọc, ví dụ Attachment dùng `text/plain`. Ngay cả khi là loại nhị phân như `image/jpeg`, dữ liệu vẫn không mã hóa lại mà dùng octet thô.
- Theo RFC 1341, boundary là chuỗi boundary định nghĩa trong header, đứng sau `<CRLF>`.

Đừng quên `<CRLF>` khi tạo hoặc parse các thông điệp này.

Header:

```
Content-Type: multipart/mixed; boundary=abcABC0123'()+_,-./:=?  
X-Experience-API-Version:1.0.0
```

Nội dung:

```
--abcABC0123'()+_,-./:=?  
Content-Type:application/json  
  
{  
  "actor": {  
    "mbox": "mailto:sample.agent@example.com",  
    "name": "Sample Agent",  
    "objectType": "Agent"  
  },  
  "verb": {  
    "id": "http://adlnet.gov/expapi/verbs/answered",  
    "display": {  
      "en-US": "answered"  
    }  
  },  
  "object": {  
    "id": "http://www.example.com/tincan/activities/multipart",  
    "objectType": "Activity",  
    "definition": {  
      "name": {  
        "en-US": "Multi Part Activity"  
      },  
      "description": {  
        "en-US": "Multi Part Activity Description"  
      }  
    }  
  }  
}
```



```

    }
  },
  "attachments": [
    {
      "usageType": "http://example.com/attachment-usage/test",
      "display": { "en-US": "A test attachment" },
      "description": { "en-US": "A test attachment (description)" },
      "contentType": "text/plain; charset=ascii",
      "length": 27,
      "sha2": "495395e777cd98da653df9615d09c0fd6bb2f8d4788394cd53c56a3bfd"
    }
  ]
}
--abcABC0123'()+_,-./:=?
Content-Type:text/plain
Content-Transfer-Encoding:binary
X-Experience-API-Hash:495395e777cd98da653df9615d09c0fd6bb2f8d4788394cd53c56a3bfd

here is a simple attachment
--abcABC0123'()+_,-./:=?--

```

#### 1.3.1.12 4.1.12 Ràng buộc dữ liệu Giải thích

Mọi thuộc tính dùng trong Statement đều bị ràng buộc kiểu dữ liệu, và các kiểu đó cũng ràng buộc hành vi của hệ thống xử lý Statement. Để làm rõ, phần này nhấn mạnh những trách nhiệm chính nơi hệ thống tuân thủ cần thực hiện hành vi cụ thể.

Yêu cầu cho client

Để nhấn mạnh, làm rõ và cung cấp hướng dẫn thực thi, các yêu cầu dưới đây lặp lại một số yêu cầu quan trọng đã xuất hiện ở phần khác. Vì việc kiểm chứng IRI đầy đủ là rất khó, phần lớn trách nhiệm đảm bảo tính khả chuyển dữ liệu nằm ở phía client.

- IRI gửi làm giá trị phải hợp lệ.
- Tương tự, key của language map phải được gửi kèm tag ngôn ngữ RFC 5646 hợp lệ (<http://tools.ietf.org/html/rfc5646>).
- Nên dùng thư viện để tạo IRI thay vì nối chuỗi.
- Trừ khi có chỉ định riêng, nên xem giá trị là phân biệt hoa/thường.
- Khi gửi dữ liệu không phân biệt hoa/thường, nên gửi ở chữ thường.
- Không nên thêm thuộc tính vào Statement trừ khi đặc tả này cho phép rõ ràng.

Ghi chú: khuyến nghị LRS từ chối Statement có thuộc tính bổ sung. Thuộc tính bổ sung trong Statement có thể khiến Statement không tương thích với một số LRS.

Yêu cầu cho LRS

- LRS phải từ chối Statement trong các trường hợp:
  - Không có giá trị (trừ bên trong extensions).
  - Nơi cần số nhưng nhận chuỗi (kể cả chuỗi chứa chữ số).
  - Nơi cần giá trị nhị phân boolean nhưng nhận chuỗi (kể cả chuỗi biểu thị giá trị nhị phân).

- Nơi cần ký tự theo định dạng đặc biệt (mailto, IRI, UUID...), nhưng nhận key hoặc value không đúng định dạng, kể cả chuỗi rỗng.
- Phân biệt hoa/thường của key không khớp với đặc tả này.
- Phân biệt hoa/thường của giá trị enum không khớp với giá trị enum trong đặc tả này.
- Key hoặc value không được đặc tả này cho phép.
- Một key xuất hiện nhiều lần trong cùng một object.
- LRS phải từ chối Statement chứa IRL/IRI không có scheme.
- Tối thiểu, LRS phải xác thực chuỗi độ dài token của language map key phù hợp chuẩn RFC 5646.
- LRS phải xử lý và ghi số với độ chính xác tối thiểu IEEE754 32-bit.
- LRS phải dùng cùng tiêu chí như kiểm tra giá trị trong Statement để kiểm tra giá trị tham số. **Ghi chú:** khác JSON, giá trị tham số chuỗi không được đặt trong dấu ngoặc kép.
- Trừ khi có chỉ định riêng, LRS nên xử lý mọi giá trị theo phân biệt hoa/thường.
- Để đáp ứng yêu cầu từ chối do sai định dạng, LRS chỉ cần best-effort validation cho định dạng IRL/IRI.
- LRS có thể dùng best-effort validation để đáp ứng yêu cầu từ chối khi language map key sai định dạng.
- Không nên thêm thuộc tính vào Statement hoặc object khác trừ khi đặc tả cho phép rõ ràng; LRS nên từ chối Statement có các thuộc tính bổ sung đó.

### 1.3.2 4.2 Truy xuất Statement

Giải thích

Có thể truy xuất một tập Statement bằng query statements. Chi tiết xem 7.2 Statement API.

Chi tiết

Bảng dưới mô tả cấu trúc dữ liệu của kết quả truy xuất từ Statement API.

```

    Nếu không có Statement khớp, thuộc tính này là mảng rỗng.
    </td>
    <td>Bắt buộc</td>
</tr>
<tr><td>more</td><td>IRL</td>
    <td>IRL tương đối dùng để lấy thêm kết quả. Bao gồm full path và query string tùy c

```

Nếu không còn kết quả để tải thêm, giá trị là chuỗi rỗng. Bắt buộc khi danh sách trả về bị giới hạn; trường hợp khác là tùy chọn

Yêu cầu

- IRL lấy từ thuộc tính more phải còn sử dụng được ít nhất 24 giờ sau khi LRS trả về.
- Để tránh phải lưu trạng thái query liên kết với IRL, LRS có thể nhúng toàn bộ thông tin cần thiết vào IRL của more.
- LRS nên tránh tạo IRL quá dài trong thuộc tính more.
- LRS có thể thực thi lại query khi IRL trong more được truy cập để trả kết quả như batch tại thời điểm query gốc chạy; có thể loại các Statement đã bị void sau đó.
- Hoặc LRS có thể cache danh sách Statement trả về trong more để khớp với những gì nên được trả tại thời điểm query gốc chạy.
- Trong trường hợp cache, LRS có thể xóa Statement đã void khỏi danh sách cache.
- Người dùng không nên suy diễn ý nghĩa từ giá trị IRL trả về trong thuộc tính more.

#### 1.3.2.1 4.3 Vô hiệu hóa Bối cảnh

Việc Statement không bị chỉnh sửa logic hoặc xóa đảm bảo độ tin cậy rằng LRS thu thập dữ liệu chính xác và đầy đủ. Tính bất biến này là một yếu tố quan trọng cho tính phân tán của xAPI.

Tuy nhiên, điều đó không có nghĩa mọi Statement sau khi phát hành đều luôn hợp lệ vĩnh viễn. Do lỗi hoặc yếu tố khác, đôi khi cần đánh dấu Statement trước đó là vô hiệu. Đây gọi là "void Statement", và reserved verb `http://adlnet.gov/expapi/verbs/voided` được dùng cho mục đích này. Không thể void chính Statement dùng để void Statement khác.

#### Yêu cầu

- Khi phát hành Statement để void Statement khác, object của Statement void phải đặt `objectType` là `StatementRef`.
- Khi phát hành Statement để void Statement khác, object của Statement void phải chỉ rõ ID của Statement cần void trong thuộc tính `id`.
- LRS chỉ đánh giá Statement là vô hiệu khi bản thân Statement đó không phải Statement void, và LRS có Statement void tham chiếu tới Statement ban đầu.
- Khi nhận Statement void Statement khác, nếu yêu cầu không đến từ nguồn có quyền void, LRS nên từ chối toàn bộ request chứa Statement void đó bằng HTTP 403 `Forbidden`.
- Khi nhận Statement void Statement khác, LRS không nên từ chối request chỉ vì object Statement cần void không tồn tại.
- Khi nhận Statement void Statement khác, LRS có thể rollback các thay đổi định nghĩa Activity hoặc Agent vốn được đưa vào bởi Statement bị void.
- Nhà cung cấp hoạt động muốn kích hoạt lại Statement đã void nên phát hành lại Statement với ID mới.
- Hệ thống báo cáo không nên hiển thị Statement đã void hoặc Statement dùng để void theo mặc định.

**Ghi chú:** Chi tiết về tạo tham chiếu đến Statement khác xem 4.1.4.3 Khi object là Statement, phần Statement Reference. Về hành vi của Statement void khi được gọi, xem `StatementRef` trong 7.2 Statement API.

#### Ví dụ

Ví dụ Statement sau void Statement trước đó có id `e05aa883-acaf-40ad-bf54-02c8ce485fb0`.

```
{
  "actor" : {
    "objectType": "Agent",
    "name" : "Example Admin",
    "mbox" : "mailto:admin@example.adlnet.gov"
  },
  "verb" : {
    "id": "http://adlnet.gov/expapi/verbs/voided",
    "display": {
      "en-US": "voided"
    }
  },
  "object" : {
    "objectType": "StatementRef",
    "id" : "e05aa883-acaf-40ad-bf54-02c8ce485fb0"
  }
}
```

### 1.3.2.2 4.4 Statement có chữ ký

**1.3.2.2.1 Giải thích** Statement có thể chứa chữ ký số để cung cấp tính xác thực và toàn vẹn mạnh, bền vững.

**1.3.2.2.2 Bối cảnh** Một số Statement có thể mang ý nghĩa quy định hoặc pháp lý. Khi đó cần bằng chứng xác thực và toàn vẹn mạnh, bền vững. Có thể cần xác minh các Statement này mà không cần tin cậy hoặc truy cập hệ thống đã ghi ban đầu. Chữ ký số cho phép hệ thống bên thứ ba xác minh các Statement như vậy.

**1.3.2.2.3 Chi tiết** Statement có chữ ký chứa JSON Web Signature (JWS) như một Attachment. Nhờ đó, bản serialize của Statement gốc có thể được đính kèm cùng chữ ký. Để tương tác liên hệ thống, nên dùng họ thuật toán "RSA+SHA" của JWS; để khả năng khám phá người ký, nên dùng chứng chỉ X.509.

#### 1.3.2.2.4 Yêu cầu

- Statement có chữ ký phải chứa JWS dưới dạng Attachment với `usageType` là `http://adlnet.gov/expapi/attachment` và `contentType` là `application/octet-stream`: `http://tools.ietf.org/html/draft-ietf-jose-json-web-signature`
- Trước khi gắn chữ ký, JWS phải có phần dữ liệu là chuỗi dữ liệu JSON hợp lệ liên tiếp của Statement được tạo.
- JWS phải dùng thuật toán R256, RS384, hoặc RS512.
- JWS nên được tạo từ private key gắn với chứng chỉ X.509.
- Nếu dùng X.509 để ký, header JWS nên chứa thuộc tính `x5c` bao gồm chuỗi chứng chỉ tương ứng.
- LRS phải từ chối bằng HTTP 400 yêu cầu ghi Statement chứa chữ ký sai định dạng.
- Phản hồi cho Statement bị từ chối nên có thông điệp giải thích.
- Để xác minh chữ ký đúng định dạng, LRS phải thực hiện:
  - Giải mã JWS và trích chuỗi Statement đã ký từ dữ liệu chữ ký JWS.
  - Chứng minh Statement gốc tương đương về mặt logic với Statement đã được LRS chấp nhận.
  - Khi kiểm tra tương đương, phải bỏ qua khác biệt do xử lý `id`, `authority`, `stored`, `timestamp`, `version` của LRS nếu đó là xử lý được phép hoặc bắt buộc.
  - Nếu header JWS chứa chứng chỉ X.509, xác thực chữ ký bằng chứng chỉ theo cách định nghĩa trong JWS.
- Client không được giả định chữ ký hợp lệ chỉ vì LRS đã chấp nhận.

**Ghi chú:** Bước xác thực với chứng chỉ X.509 kèm theo không nhằm mục đích bảo mật, mà để phát hiện lỗi trong chữ ký. Các bước xác thực Statement có chữ ký thay đổi tùy mức độ đảm bảo yêu cầu và nằm ngoài phạm vi đặc tả này.

**1.3.2.2.5 Ví dụ** Xem ví dụ tại Appendix E: Ví dụ Statement có chữ ký.

## 1.4 5.0 Các kiểu dữ liệu khác

### 1.4.1 5.1 Tài liệu

Giải thích

xAPI cung cấp cho nhà cung cấp hoạt động khả năng ghi dữ liệu tùy ý ở dạng tài liệu cho Activity, Agent, hoặc cả hai.

Chi tiết

Lưu ý bảng dưới khác các bảng khác trong đặc tả này: nó mô tả thuộc tính tổng quát thay vì JSON object. `ID` được chỉ định bằng IRL, `updated` được chỉ định bằng HTTP header, còn `contents` là chính tài liệu HTTP (không phải object).

## 1.4.2 5.2 Bản đồ ngôn ngữ

### Giải thích

Language Map là từ điển lấy RFC 5646 language tag làm key, và giá trị là chuỗi trong ngôn ngữ được tag chỉ định. Map này nên được làm phong phú tối đa trong phạm vi hiểu biết về chuỗi đó ở các ngôn ngữ mục tiêu.

Chuỗi nội dung trong Language Map là văn bản thuần. Không giả định xử lý các định dạng như HTML tag hay Markdown; khi hiển thị cho người dùng cuối, chúng sẽ được hiển thị nguyên văn dạng mã. Ngoại lệ là khi Language Map được dùng trong Extension và chủ sở hữu Extension IRI chỉ rõ cơ chế hiển thị cụ thể.

## 1.4.3 5.3 Mở rộng

### Giải thích

Extensions có thể được dùng như một phần của Activity Definition, context của Statement, hoặc result của Statement. Trong mỗi trường hợp, chúng cung cấp cách tự nhiên để mở rộng các thành phần cho mục đích sử dụng cụ thể. Nội dung Extension có thể phục vụ một ứng dụng riêng lẻ hoặc là quy ước cho cả cộng đồng thực hành.

### Chi tiết

Extension được định nghĩa bằng map, có liên hệ logic với phần tương ứng của Statement nơi nó xuất hiện. Giá trị Extension là JSON value hoặc cấu trúc dữ liệu. Extension trong context cung cấp ngữ cảnh cho trải nghiệm trung tâm; còn trong result cung cấp yếu tố liên quan kết quả. Trong Activity, Extension cung cấp thông tin bổ sung hỗ trợ định nghĩa Activity trong một ứng dụng tùy chỉnh hoặc trong cộng đồng cụ thể. Ý nghĩa và cấu trúc giá trị của Extension dưới một IRI key được định nghĩa bởi bên quản trị IRI đó.

### Yêu cầu

- Key của extension map phải là IRI.
- LRS không được từ chối Statement chỉ dựa trên giá trị extension map.
- Client luôn nên nỗ lực map càng nhiều thông tin càng tốt vào các phần tử có sẵn để tăng khả năng tương tác giữa các công cụ tuân thủ xAPI.
- Mọi extension IRI nên có controller.
- Controller của IRL extension key nên cung cấp mô tả dễ đọc cho con người về ý nghĩa Extension ngay tại IRL.

**Ghi chú:** Statement chỉ gồm Extensions là vô nghĩa vì hệ thống khác không thể hiểu ý nghĩa.

## 1.4.4 5.4 Siêu dữ liệu định danh

**1.4.4.0.1 Giải thích** Có thể lưu thông tin bổ sung về định danh trong Statement và tại vị trí do định danh IRI chỉ tới. Việc chứa metadata trong Statement cho phép biểu diễn metadata của IRI mà không cần resolve IRI. Việc lưu metadata tại vị trí IRI cho phép chủ sở hữu IRI định nghĩa metadata chuẩn cho IRI đó.

**1.4.4.0.2 Chi tiết** Trong đặc tả này có một số loại định danh IRI:

- Verb
- Activity ID
- Activity Type
- Extension Key
- Attachment Usage Type

Về cấu trúc metadata lưu cho Activity ID, xem Activity Definition Object.

Về cấu trúc metadata lưu cho các định danh khác, xem định dạng dưới:

Metadata lưu trữ là tài liệu chứa JSON object nêu trên. Khi metadata được cung cấp, nó trở thành nguồn chuẩn cho thông tin mà định danh biểu thị. Ngoài Activity ID, nhà cung cấp hoạt động được khuyến nghị dùng các định danh IRI đã được thiết lập và thừa nhận rộng rãi cho mọi loại định danh.

#### 1.4.4.0.3 Yêu cầu

- Có thể cung cấp metadata cho định danh.
- Khi cung cấp metadata, nên có cả name và description.
- IRL nên được định nghĩa trong miền do người tạo IRL quản lý.
- Với các IRI dùng làm định danh nêu trên, nếu IRI là IRL được tạo theo đặc tả này, quản trị IRL đó nên cung cấp JSON metadata này tại vị trí IRL khi được yêu cầu với Content-Type: application/json.
- Khi định danh đã tồn tại, nhà cung cấp hoạt động nên dùng định danh liên quan đã có.
- Nếu không có định danh phù hợp, nhà cung cấp hoạt động có thể tạo và dùng định danh riêng.
- Khi định nghĩa định danh, nhà cung cấp hoạt động có thể dùng nhóm URI có anchor để một trang có thể chứa định nghĩa của nhiều định danh. Ví dụ: `http://example.com/xapi/verbs#defenestrated`
- Có thể dùng nguồn thông tin khác để bổ sung chi tiết còn thiếu (như dịch thuật), hoặc thay thế hoàn toàn metadata này nếu metadata không có hoặc không đọc được; đặc biệt khi định danh không được tạo cho mục đích dùng trong đặc tả này.

## 1.5 6.0 Giao tiếp lúc chạy

Trong Chương 6 và 7, tài liệu đi sâu khía cạnh kỹ thuật của xAPI đồng thời mô tả cách Statement được truyền giữa nhà cung cấp hoạt động và LRS. Hiện có một số thư viện đang phát triển cho tập kỹ thuật xử lý phần đặc tả ở chương này (bao gồm JavaScript), vì vậy người phát triển nội dung không nhất thiết phải hiểu toàn bộ chi tiết của chương này.

### 1.5.1 6.1 Mã hóa

Yêu cầu

- Mọi chuỗi ký tự phải được mã hóa và diễn giải theo UTF-8.

### 1.5.2 6.2 Quản lý phiên bản API

Bối cảnh

Trong các bản sửa đổi tương lai của đặc tả, có thể có thay đổi như thêm thuộc tính vào Statement.

Do đó, hệ thống đọc Statement có thể nhận phản hồi chứa Statement ở các phiên bản khác nhau. Sự khác biệt phiên bản này có thể được xử lý chính xác hơn bằng version header, đồng thời giúp bảo đảm không tồn tại triển khai phiên bản chưa hoàn chỉnh hoặc bị trộn lẫn.

Nhờ sử dụng Semantic Versioning, client và LRS có thể biết chắc tính tương thích ngay cả khi đặc tả thay đổi.

Chi tiết

Từ phiên bản 1.0.0, xAPI được đánh version theo Semantic Versioning 1.0.0. Mọi request từ client và mọi response từ LRS đều phải có HTTP header với tên X-Experience-API-Version và giá trị là version.

Ví dụ: X-Experience-API-Version: 1.0.1

Với version 1.0.3 thì header là `X-Experience-API-Version: 1.0.3`. Xem Revision History cho phiên bản của đặc tả này.

#### Yêu cầu cho LRS

- LRS phải bao gồm header `X-Experience-API-Version` trong mọi response.
- LRS phải đặt giá trị mới nhất vào header này.
- LRS phải chấp nhận request có version header `1.0` và coi như `1.0.0`.
- LRS phải từ chối request có version header trước `1.0.0`, trừ khi request đó được chuyển tới triển khai hoàn toàn tuân thủ phiên bản cũ đã chỉ định.
- LRS phải từ chối request có version header `1.1.0` hoặc cao hơn.
- Khi từ chối, LRS phải phản hồi HTTP 400 kèm mô tả ngắn về vấn đề.

#### Yêu cầu cho client

- Client phải bao gồm header `X-Experience-API-Version` trong mọi request.
- Client phải đặt giá trị mới nhất vào header này.
- Client nên chấp nhận response có version `1.0.0` hoặc cao hơn.
- Client nên chấp nhận cấu trúc dữ liệu có thuộc tính được thêm mới.
- Client nên bỏ qua mọi thuộc tính không được quy định trong đặc tả version `1.0.0`.

#### Yêu cầu về chuyển đổi

- Hệ thống không được chuyển Statement phiên bản mới sang định dạng cũ chỉ để xử lý khác biệt phiên bản.
- Hệ thống chỉ được chuyển Statement phiên bản cũ sang định dạng mới khi tuân theo phương pháp trong Appendix D: Chuyển Statement về 1.0.0.

### 1.5.3 6.3 Đồng thời (Concurrency)

**1.5.3.0.1 Giải thích** Kiểm soát đồng thời đảm bảo người dùng API không PUT/POST/DELETE lên LRS dựa trên dữ liệu lỗi thời.

**1.5.3.0.2 Chi tiết** xAPI dùng HTTP1.1 entity tags (ETags) để triển khai optimistic concurrency control cho phần API mà PUT/POST/DELETE có thể ghi đè hoặc xóa thực thể dữ liệu hiện có, gồm:

- State API
- Agent Profile API
- Activity Profile API

Do xung đột trạng thái hiếm xảy ra, State API cho phép PUT/POST/DELETE không có concurrency header. Các yêu cầu dưới đây chỉ áp dụng cho Agent Profile API và Activity Profile API.

#### 1.5.3.0.3 Yêu cầu cho client

- Client gửi PUT tới Agent Profile API hoặc Activity Profile API phải có header `If-Match` hoặc `If-None-Match`.
- Client gửi POST tới Agent Profile API hoặc Activity Profile API nên có header `If-Match` hoặc `If-None-Match`.
- Client gửi DELETE tới Agent Profile API hoặc Activity Profile API nên có header `If-Match`.

#### 1.5.3.0.4 Yêu cầu cho LRS

- Khi phản hồi GET, LRS phải thêm header HTTP `ETag` vào response. (Mục đích là để người dùng API không đọc được ETag header vẫn có thể tự tính giá trị này.)

- Khi phản hồi PUT/POST, LRS nên đính kèm header HTTP ETag cho thực thể được tạo hoặc cập nhật.
- Khi phản hồi GET, LRS phải tính giá trị header là chuỗi hex của SHA-1 digest của phần nội dung.
- Khi phản hồi GET, LRS phải đặt giá trị header trong dấu ngoặc kép.
- Khi phản hồi PUT, nếu có If-Match kèm ETag, LRS phải xử lý header If-Match theo RFC2616 (HTTP1.1) để phát hiện thay đổi sau lần đọc gần nhất của người dùng.
- Khi phản hồi PUT, nếu có If-None-Match (bao gồm "\*"), LRS phải xử lý theo RFC2616 để phát hiện có tài nguyên mà người dùng không biết.
- Khi phản hồi POST/DELETE, nếu có ETag, LRS nên xử lý If-Match theo RFC2616 để người dùng phát hiện thay đổi kể từ lần đọc trước.
- Khi phản hồi POST, nếu có "\*", LRS nên xử lý If-None-Match theo RFC2616 để phát hiện tài nguyên tồn tại mà người dùng không biết.

Nếu với PUT mà điều kiện tiên quyết trong header không thỏa, LRS phải:

- Trả HTTP 412 Precondition Failed.
- Không thay đổi tài nguyên.

Nếu với POST hoặc DELETE mà điều kiện tiên quyết trong header không thỏa, LRS nên:

- Trả HTTP 412 Precondition Failed.
- Không thay đổi tài nguyên.

Nếu nhận PUT lên tài nguyên đã tồn tại mà không có cả hai header nêu trên, LRS phải:

- Trả HTTP 409 Conflict.
- Trả response giải thích người dùng cần:
  - Kiểm tra trạng thái hiện tại của tài nguyên.
  - Gửi If-Match với ETag hiện tại để giải quyết xung đột.
- Không thay đổi tài nguyên.

## 1.5.4 6.4 Bảo mật

Bối cảnh

Đề cân bằng giữa khả năng tương tác và các yêu cầu bảo mật khác nhau ở nhiều môi trường, đặc tả định nghĩa một số tùy chọn xác thực.

Chi tiết

Bảng dưới mô tả các kịch bản xác thực giả định.

**Ứng dụng đã đăng ký** là ứng dụng đã được xác thực với LRS như một OAuth consumer đã đăng ký sẵn trong LRS.

**Người dùng đã được nhận diện** là tài khoản người dùng trên LRS hoặc trên hệ thống mà LRS ủy quyền định nghĩa người dùng.

Yêu cầu

LRS phải hỗ trợ xác thực bằng ít nhất một trong các cách sau:

- OAuth 1.0 (RFC 5849) (với phương thức ký HMAC-SHA1, RSA-SHA1, PLAIN TEXT)
- HTTP Basic Auth
- Common Access Cards (triển khai sẽ được theo dõi và chi tiết hóa trong phiên bản tương lai)
- LRS phải quyết định hoặc ủy quyền quyết định về tính hợp lệ của Statement và xác định xử lý phù hợp dựa trên thông tin xác thực đã dùng.



#### 1.5.4.1 6.4.1 Quy trình cho từng kịch bản

##### 1.5.4.1.1 Yêu cầu

- LRS phải ghi nhận tên ứng dụng và consumer key (định danh) duy nhất.
- LRS phải cung cấp cơ chế hoàn tất đăng ký này hoặc ủy thác cho hệ thống khác có cơ chế tương đương.
- Để hỗ trợ xAPI đầy đủ, LRS phải cấu hình được:
  - Theo một trong các workflow bên dưới.
  - Theo một trong các workflow scenario bên dưới.
- Vì lý do bảo mật, LRS có thể:
  - Chỉ hỗ trợ tập con các workflow bên dưới.
  - Giới hạn người dùng đã nhận diện hoặc ứng dụng đã đăng ký.
- Tối thiểu LRS nên cung cấp OAuth ký bằng HMAC-SHA1 và RSA-SHA1.

##### 1.5.4.1.2 Quy trình và yêu cầu cho: ứng dụng đã đăng ký + người dùng đã nhận diện

- Dùng endpoint trong 6.4.2 OAuth Scope để hoàn tất workflow OAuth chuẩn (không đi vào chi tiết ở đây).
- Nếu hình thức xác thực này được dùng để ghi Statement mà không chỉ định authority, LRS nên ghi authority là Group gồm Agent biểu diễn ứng dụng đã đăng ký và Agent biểu diễn người dùng đã nhận diện.

##### 1.5.4.1.3 Quy trình và yêu cầu cho: ứng dụng đã đăng ký + người dùng chưa nhận diện

- LRS sẽ nhận request ký bằng OAuth với chứng thực của ứng dụng đã đăng ký cùng token/token secret rỗng.
- Nếu hình thức xác thực này được dùng để ghi Statement mà không chỉ định authority, LRS nên ghi authority là Agent biểu diễn ứng dụng đã đăng ký.

##### 1.5.4.1.4 Quy trình và yêu cầu cho: ứng dụng chưa đăng ký + người dùng đã nhận diện

- AP phải dùng consumer secret chứa chuỗi rỗng.
- Gọi request Temporary Credential.
- Thiết lập consumer name và các tham số chung khác. Khi đó người dùng sẽ thấy cảnh báo rằng không thể xác minh consumer name và định danh ứng dụng yêu cầu xác thực.
- Vì OAuth chỉ ra ứng dụng, LRS phải ghi authority bao gồm cả ứng dụng và người dùng đã xác thực dưới dạng Group.

##### 1.5.4.1.5 Quy trình và yêu cầu cho: không có ứng dụng + người dùng đã nhận diện

- Dùng cặp username/password tương ứng đăng nhập LRS.
- Authority được ghi là Agent định danh qua đăng nhập, trừ khi:
  - Có chỉ định thông tin authority khác, và
  - LRS tin cậy người dùng đã nhận diện đang chỉ định authority đó.

##### 1.5.4.1.6 Quy trình và yêu cầu cho: không xác thực

- Request phải có header HTTP Basic Auth với username/password là ký tự khoảng trắng.
- Request nên có header HTTP Basic Auth mà cả username/password đều là chuỗi rỗng. Trong trường hợp này, header có dạng Basic theo sau bởi giá trị base64 của :, tức chuỗi Basic Og==.

Điều này để phân biệt request đáng lẽ phải nhận challenge HTTP Basic Auth với request không được cấp quyền rõ ràng.

### 1.5.4.2 6.4.2 OAuth Scope

**1.5.4.2.1 Giải thích** Đưa ra khuyến nghị về scope để ứng dụng giao tiếp với LRS qua xAPI có thể điều chỉnh mức truy cập đáp ứng nhu cầu ứng dụng nhưng giảm thiểu khả năng lạm dụng. Giới hạn của từng scope được cộng thêm vào bất kỳ giới hạn bảo mật nào áp dụng cho tài khoản người dùng liên quan request.

**1.5.4.2.2 Chi tiết** Bảng dưới liệt kê các giá trị scope xAPI.

OAuth extension parameters

Lưu ý các tham số như `consumer_name` và `scope` không thuộc OAuth1.0. Vì vậy, nếu dùng, chúng không được chuyển như OAuth header mà phải được truyền như query string hoặc form parameter.

OAuth endpoints

Ví dụ

Danh sách scope quyết định tập quyền được yêu cầu. Ví dụ, giảng viên có thể cấp `statements/read` cho công cụ báo cáo. Tuy nhiên, LRS có thể giới hạn công cụ chỉ đọc các Statement mà giảng viên đó đọc được khi truy vấn trực tiếp vào LRS (ví dụ Statement của học viên của chính giảng viên đó).

### 1.5.4.2.3 Yêu cầu

- LRS phải chấp nhận scope parameter định nghĩa trong OAuth 2.0.
- Nếu không chỉ định scope, LRS phải giả định request scope là `statements/write` và `statements/read/mine`.
- Tối thiểu LRS phải hỗ trợ scope `all`.
- LRS có thể hỗ trợ các scope khác.
- Client nên chỉ yêu cầu scope tối thiểu cần thiết để tăng khả năng request được chấp nhận.

## 1.6 7.0 Truyền dữ liệu (REST)

Phần này mô tả 4 sub-API cấu thành xAPI (Statement, State, Agent Profile, Activity Profile). 4 sub-API được xử lý bằng các HTTP method theo phong cách RESTful. Statement API có thể được dùng độc lập để theo dõi hồ sơ học tập.

**Ghi chú:** Trong các ví dụ endpoint của đặc tả này, `http://example.com/xAPI/` được dùng làm ví dụ base endpoint của LRS. IRI theo sau cho biết endpoint cụ thể được dùng. Danh sách đầy đủ endpoint xem Appendix F: Table of All Endpoints.

Yêu cầu

- LRS phải hỗ trợ mọi endpoint nêu trong chương này.
- Nếu LRS triển khai OAuth 1.0, LRS phải hỗ trợ mọi OAuth endpoint nêu trong 6.4.2 OAuth Authorization Scope.

### 1.6.1 7.1 Mã lỗi

Giải thích

Đặc tả này định nghĩa yêu cầu để LRS chấp nhận/từ chối request, trả response và các hành vi khác trong điều kiện cụ thể. Khi LRS cần từ chối request, một phần yêu cầu là phải trả mã lỗi phù hợp.

Những yêu cầu này không mâu thuẫn với khả năng cấu hình LRS từ chối request và response theo cách khác dựa trên điều kiện ngoài phạm vi đặc tả này. Một trong các điều kiện đó là permission.

Ví dụ, LRS có thể gán permission cho một tổ hợp điều kiện sao cho chỉ một số Agent nhất định được phép, và có thể từ chối Statement dùng điều kiện không liên quan Agent. Các permission LRS có thể cấp nằm ngoài phạm vi đặc tả này, ngoại trừ danh sách giá trị OAuth scope khuyến nghị tại 6.4.2.

Permission cũng ảnh hưởng response của LRS cho GET request. Ví dụ, một tập điều kiện có thể chỉ cho phép xem Statement của một Actor cụ thể, khi đó LRS sẽ lọc Statement không liên quan Actor đó. Chi tiết xem Section 7.2.3 GET Statements.

Khi đặc tả này cho phép rõ ràng, điều kiện chỉ định cũng có thể ảnh hưởng hành vi xử lý request của LRS. Ví dụ, thông thường LRS ghi đề phần Authority của Statement, nhưng có thể chấp nhận authority nhận vào nếu có quan hệ tin cậy mạnh đối với thông tin xác thực dùng để phát Statement. Chi tiết xem Section 4.1.9 Authority.

Permission do LRS đặt có thể khiến một LRS về kỹ thuật vẫn tuân thủ nhưng trượt bài test conformance, nếu request do công cụ test phát bị từ chối bởi permission. Vì vậy, LRS cần có khả năng cấu hình để hạn chế permission không ảnh hưởng kết quả conformance test, và thông tin xác thực dùng để test cần được cấp permission phù hợp.

Điều kiện khác là request gửi tới vượt giới hạn kích thước do LRS đặt. Không hợp lý khi kỳ vọng LRS luôn nhận request mọi kích cỡ. LRS có thể chọn giới hạn kích cỡ tùy nhu cầu, nhưng khi conformance test phải có khả năng cấu hình để tránh vướng giới hạn. Tất nhiên do ràng buộc phần cứng vẫn có thể có giới hạn ngay cả trong test, nhưng giới hạn đó được kỳ vọng đủ cao để không cản trở việc chạy test.

LRS có thể từ chối request hoặc thu hồi chứng thực nếu nghi ngờ bất thường, ví dụ nhận lượng lớn request trong thời gian ngắn. Trong thời gian conformance test, các giới hạn kiểu này được kỳ vọng đặt đủ cao để không chặn lưu lượng test.

#### Chi tiết

Danh sách dưới là hướng dẫn chung về HTTP error code trả về bởi các phương thức API.

- 400 Bad Request - lỗi do tham số không hợp lệ hoặc thiếu. "Không hợp lệ" gồm JSON sai định dạng hoặc cấu trúc object không hợp lệ.
- 401 Unauthorized - cần xác thực hoặc thông tin xác thực trong request bị từ chối.
- 403 Forbidden - request bị từ chối với thông tin xác thực đã cấp. Khác với việc bản thân thông tin xác thực bị từ chối; ở đây thông tin đã được xác minh nhưng client không được phép thực hiện hành động.
- 404 Not Found - không tìm thấy tài nguyên yêu cầu. Ví dụ: gọi State/Agent Profile/Activity Profile API nhắm tài liệu cụ thể, hoặc phương thức trả về một Statement đơn.
- 409 Conflict - lỗi do xung đột với trạng thái hiện tại của tài nguyên khi gọi State API, Agent Profile API, Activity Profile API, hoặc Statement PUT/POST. Xem 6.3 Concurrency.
- 412 Precondition Failed - lỗi do điều kiện tiên quyết trong request thất bại khi gọi State API, Agent Profile API, Activity Profile API. Xem 6.3 Concurrency.
- 413 Request Entity Too Large - LRS từ chối Statement hoặc tài liệu vì kích thước vượt tối đa cho phép (hoặc Attachment trong request quá lớn).
- 429 Too Many Requests - LRS từ chối request vì nhận quá nhiều request từ client trong một khoảng thời gian.
- 500 Internal Server Error - lỗi chung, ví dụ có ngoại lệ không mong muốn trên server.

#### Yêu cầu

- LRS phải trả mã lỗi phù hợp nhất cho trạng thái lỗi trong danh sách trên.
- LRS nên dùng content negotiation nêu tại RFC 7231 để quyết định format lỗi.
- LRS nên cho phép response lỗi dạng plain text, HTML, và JSON (dùng content negotiation).
- AP nên gửi header Accept trong request để cho phép content negotiation.
- LRS nên trả message giải thích nguyên nhân lỗi trong response.

- LRS phải từ chối bằng HTTP 400 Bad Request request tới API dùng tham số mà LRS không nhận diện được trong ngữ cảnh dự kiến theo đặc tả này. (Lưu ý: LRS có thể nhận diện hành vi cho tham số không thuộc đặc tả.)
- LRS phải từ chối bằng HTTP 400 Bad Request request tới API dùng tham số khớp tên tham số trong đặc tả này nhưng giá trị không hợp lệ theo đặc tả.
- Nếu Statement trong batch bị từ chối, LRS phải từ chối toàn bộ batch.
- Nếu thông tin xác thực tương ứng request không có permission thực hiện request đó, LRS phải từ chối bằng HTTP 403 Forbidden.
- Request chứa Attachment, Statement, hoặc tài liệu lớn hơn giới hạn cho phép phải bị LRS từ chối bằng HTTP 413 Request Entity Too Large.
- LRS có thể đặt giới hạn kích thước cho Attachment/Statement/tài liệu và có thể thay đổi giới hạn theo nhiều tiêu chí, ví dụ theo permission.
- Nếu LRS từ chối request vì nhận quá nhiều request từ một client/chứng thực trong thời gian nhất định, LRS phải từ chối bằng HTTP 429 Too Many Requests.
- LRS có thể đặt giới hạn tốc độ nhận request và thay đổi theo nhiều tiêu chí, ví dụ theo permission.

Các yêu cầu dưới đây phục vụ conformance test, nhằm bảo đảm giới hạn hoặc permission do LRS đặt không cản trở test.

- LRS nên có thể cấu hình để không từ chối request từ một bộ chứng thực cụ thể vì lý do permission. Bộ chứng thực này dùng cho conformance test và có thể bị xóa/vô hiệu trong môi trường thật.
- LRS phải có thể cấu hình để chấp nhận Attachment, Statement hoặc tài liệu với kích thước hợp lý (như nêu trên).
- LRS phải có thể cấu hình để chấp nhận request với tốc độ gửi hợp lý.

## 1.6.2 7.2 Statement API

Giải thích

Đây là phương thức giao tiếp cốt lõi của Experience API.

### 1.6.2.1 7.2.1 PUT Statement Chi tiết

Ví dụ endpoint: `http://example.com/xAPI/statements` Lưu một Statement đơn với ID đã cho. (Lưu Statement đơn cũng có thể dùng POST.)

Giá trị trả về: 204 No Content

Yêu cầu cho LRS

- Kể cả khi nhận Statement có Statement ID đã được gán trước đó, LRS không được thay đổi bất kỳ trạng thái nào. Dù phản hồi 409 Conflict hay 204 No Content, không được thay đổi Statement hay bất kỳ object nào khác.
- Nếu Statement ID nhận vào đã tồn tại trên LRS, LRS nên kiểm tra Statement nhận vào có khớp bản đã có hay không; nếu không khớp thì trả 409 Conflict.
- LRS có thể phản hồi trước khi Statement đã lưu trở nên truy xuất được.

Yêu cầu cho nhà cung cấp hoạt động

- Nhà cung cấp hoạt động nên POST (thay vì PUT) Statement có thuộc tính `id`.
- Nếu PUT Statement, nên dùng thuộc tính `id` của Statement.
- Nếu được cung cấp, thuộc tính `id` của Statement phải khớp tham số `statementId` trong request.

### 1.6.2.2 7.2.2 POST Statement Chi tiết

Ví dụ endpoint: `http://example.com/xAPI/statements`

Lưu một Statement hoặc một nhóm Statement. Một phương án thay thế cho hệ thống tạo khối lượng lớn Statement là cung cấp API phía AP để LRS định kỳ gọi lấy dữ liệu mới/cập nhật. Cách này chỉ thực tế với hệ thống cung cấp dữ liệu lớn cho LRS.

Giá trị trả về: 200 OK, mảng Statement id(s) (UUID).

Yêu cầu

Kể cả khi nhận Statement có Statement ID đã được gán trước đó, LRS không được thay đổi trạng thái nào. Dù phản hồi 409 Conflict hay 200 OK, không được thay đổi Statement hay bất kỳ object nào khác.

Nếu Statement ID nhận vào đã tồn tại trên LRS, LRS nên kiểm tra Statement nhận vào có khớp bản đã có hay không; nếu không khớp thì trả 409 Conflict.

LRS có thể phản hồi trước khi Statement đã lưu trở nên truy xuất được.

- GET Statements có thể được gọi dưới dạng POST với form property trong query string khi cần, nhưng có ràng buộc. Xem 7.8 Cross-Origin Requests.
- LRS phải có khả năng phân biệt POST dùng để thêm Statement đơn hay danh sách nhiều Statement qua tham số truyền vào. Xem 7.8 Cross-Origin Requests.

### 7.2.3 GET Statement

Chi tiết

Ví dụ endpoint: `http://example.com/xAPI/statements`

Phương thức này dùng để lấy một Statement đơn hoặc nhiều Statement. Nếu chỉ định `statementId` hoặc `voidedStatementId`, phản hồi trả về một Statement đơn.

Ngược lại, phản hồi là Statement Result: danh sách Statement được sắp theo `stored` mới trước cũ, trong giới hạn permission và độ dài danh sách tối đa. Nếu có thêm kết quả, IRL để lấy tiếp nằm trong StatementResult object.

Giá trị trả về: 200 OK, Statement hoặc Statement Result (xem 4.2).

**Lưu ý:** Giá trị tham số boolean giống JSON: `true` hoặc `false`.

Yêu cầu

- Nếu request tới resource này chứa cả `statementId` và `voidedStatementId`, LRS phải từ chối với lỗi HTTP 400.
- Nếu request chứa `statementId` hoặc `voidedStatementId` đồng thời có tham số khác ngoài `attachments` hoặc `format`, LRS phải từ chối với lỗi HTTP 400.
- LRS có thể áp dụng điều kiện lọc query dựa trên permission gắn với thông tin xác thực đang dùng.
- Ngay cả khi không có Statement khớp điều kiện lọc, LRS vẫn phải trả HTTP 200 cùng StatementResult object; khi đó `statements` là mảng rỗng.
- Với mọi response cho Statement request, LRS phải có header `X-Experience-API-Consistent-Through` ở định dạng ISO 8601 combined date and time. Giá trị này là timestamp cho biết mọi Statement có `stored` trước thời điểm đó đều đã có thể truy xuất với độ tin cậy hợp lý. Thời điểm này cần tính tới trạng thái tạm thời như quá tải có thể gây trễ khả năng truy xuất.
- Nếu GET Statement có `attachments=true`, LRS phải dùng multipart response format và phải gồm toàn bộ Attachments như mô tả ở 4.1.11.

- Nếu GET Statement có `attachments=false`, LRS không được gồm attachment raw data và phải trả theo `application/json`.

Điều kiện lọc cho StatementRef

Khi tham số lọc không dựa trên thời gian/chuỗi (tức ngoài `since`, `until`, `limit`), nếu một Statement tham chiếu Statement khác bằng StatementRef ở object, thì nếu Statement đích thỏa điều kiện lọc, Statement nguồn cũng thỏa điều kiện lọc. Các tham số dựa trên thời gian/chuỗi vẫn cần áp dụng lên chính Statement chứa StatementRef. Quy tắc này áp dụng đệ quy: nếu A tham chiếu B, B tham chiếu C, và C thỏa điều kiện lọc nêu trên thì A cũng là đối tượng lọc.

Ví dụ có Statement "Ben đã đỗ khóa huấn luyện chất nổ" và Statement follow-up "Andrew đã xác nhận <StatementRef to original Statement>". Dù follow-up không nhắc trực tiếp "Ben" hay "khóa huấn luyện chất nổ", nếu lọc theo `actor="Ben"` hoặc `activity="khóa huấn luyện chất nổ"`, cả hai Statement đều sẽ khớp và được trả (miễn đáp ứng điều kiện thời gian/chuỗi).

Các điều kiện này không áp dụng khi tìm Statement theo `statementId` hoặc `voidedStatementId`.

**Ghi chú:** StatementRef nằm trong thuộc tính `statement` của context không ảnh hưởng tới bộ lọc Statement.

- Với Activity object có Language Map ở tên/mô tả/thành phần interaction: LRS phải trả về một ngôn ngữ cho từng map đó.
- Với Verb object có Language Map ở `display`: LRS nên trả duy nhất một ngôn ngữ trong map đó.
- Để chọn ngôn ngữ phù hợp nhất, LRS phải áp dụng header `Accept-Language` theo RFC 2616 (HTTP 1.1). Lưu ý việc chọn entry ngôn ngữ phải áp dụng theo từng Language Map, không áp dụng cho toàn bộ resource (danh sách Statement) như một khối.

#### 7.2.4 Statement đã vô hiệu

Yêu cầu

- LRS không được trả Statement đã void, trừ khi Statement đó được yêu cầu bằng `voidedStatementId`.
- Theo điều kiện lọc bằng StatementRef, khi truy xuất trực tiếp/gián tiếp theo tiêu chí thời gian hoặc chuỗi, LRS phải trả cả Statement tham chiếu Statement đã void (trừ khi chính Statement tham chiếu đó cũng bị void). Điều này gồm cả Statement void (vốn không thể tự void chính nó). Công cụ báo cáo có thể dùng `target` của Statement void để nhận diện sự tồn tại và ID của mọi Statement void.
- Công cụ báo cáo nên dùng `voidedStatementId` để request riêng từng Statement đã void.

### 1.6.3 7.3 Document API

Ba Document API cung cấp lưu trữ tài liệu cho Activity Provider và Agent. Chi tiết từng API nằm ở các mục kế tiếp; nội dung trong mục này áp dụng cho cả 3 API.

Chi tiết

Yêu cầu

- Activity Provider có thể gửi tài liệu về Activity/Agent mà LRS chưa có tri thức trước đó tới bất kỳ Document API nào.
- LRS không được từ chối tài liệu chỉ vì chưa có tri thức trước về Activity và/hoặc Agent đó.

Yêu cầu về quy trình JSON

Nếu Activity Provider lưu biến dưới dạng JSON object, với tài liệu `Content-Type: application/json`, có thể dùng POST để xử lý chúng như một tập biến.

Ví dụ tài liệu chứa:

```
{
  "x" : "foo",
  "y" : "bar"
}
```

Nếu LRS nhận POST Content-Type: application/json lên tài liệu hiện có cũng có Content-Type: application/json, LRS phải merge tài liệu được post vào tài liệu hiện có. Trong ngữ cảnh này, "merge" nghĩa là:

- Deserialize object từ nhiều tài liệu.
- Với mỗi thuộc tính được định nghĩa trực tiếp trong object được post, gán giá trị tương ứng lên object hiện có.
- Lưu serialization JSON hợp lệ của object hiện có làm tài liệu tham chiếu cho request.

Lưu ý chỉ merge thuộc tính ở cấp cao nhất. Kể cả khi giá trị thuộc tính cấp cao là object, toàn bộ giá trị cũ của thuộc tính đó sẽ bị thay thế hoàn toàn bởi giá trị mới.

Ví dụ nếu POST lại với cùng ID tài liệu như trên bằng:

```
{
  "x" : "bash",
  "z" : "faz"
}
```

Tài liệu lưu trong LRS sẽ là:

```
{
  "x" : "bash",
  "y" : "bar",
  "z" : "faz"
}
```

Nếu tài liệu gốc tồn tại và content type của tài liệu gốc hoặc tài liệu post không phải application/json, hoặc một trong hai không parse được thành JSON object, LRS phải trả HTTP 400 Bad Request và không được cập nhật tài liệu đích.

Nếu tài liệu gốc chưa tồn tại, LRS phải xử lý như PUT và lưu tài liệu được POST. Nếu merge thành công, LRS phải trả HTTP 204 No Content.

Nếu AP cần xóa thuộc tính, nên dùng PUT để thay toàn bộ tài liệu như mô tả sau.

## 1.6.4 7.4 State API

**1.6.4.0.1 Giải thích** Về cơ bản đây là scratch area dùng khi Activity Provider không có kho lưu trữ nội bộ, hoặc khi cần chia sẻ trạng thái giữa nhiều thiết bị.

**1.6.4.0.2 Chi tiết** Ý nghĩa cuộc gọi do tham số stateId quyết định. Nếu có stateId, phương thức GET/DELETE thao tác trên một tài liệu state đơn được chỉ định bởi stateId. Nếu không có stateId, GET trả các ID khả dụng, còn DELETE xóa toàn bộ state trong ngữ cảnh do các tham số khác xác định.

Tài liệu đơn (PUT | POST | GET | DELETE)

Ví dụ endpoint: `http://example.com/xAPI/activities/state`

Ghi, sửa, lấy hoặc xóa tài liệu được chỉ định bởi stateId trong ngữ cảnh gồm activity, agent và registration (nếu có).

Giá trị trả về (PUT | POST | DELETE): 204 No Content

Giá trị trả về (GET): 200 OK, State Content

GET nhiều tài liệu

Ví dụ endpoint: `http://example.com/xAPI/activities/state`

Lấy ID của toàn bộ state data trong ngữ cảnh này (activity + agent [+ registration nếu chỉ định]). Nếu có tham số since, kết quả chỉ gồm entry được ghi hoặc cập nhật sau timestamp chỉ định (không gồm chính thời điểm đó).

Giá trị trả về: 200 OK, Array of ids

DELETE nhiều tài liệu

Ví dụ endpoint: `http://example.com/xAPI/activities/state`

Xóa toàn bộ state data trong ngữ cảnh này (activity + agent [+ registration nếu chỉ định]).

Giá trị trả về: 204 No Content

### 1.6.5 7.5 Activity Profile API

**1.6.5.0.1 Giải thích** Activity Profile API tương tự State API, cho phép lưu cặp key/document tùy ý gắn với Activity.

**1.6.5.0.2 Chi tiết** Ý nghĩa cuộc gọi do tham số profileId quyết định. Nếu có profileId, GET thao tác trên một tài liệu đơn xác định bởi profileId. Nếu không có, GET trả về các id khả dụng.

Activity Profile API cũng có phương thức đọc toàn bộ mô tả Activity từ LRS.

GET toàn bộ Activity object

Ví dụ endpoint: `http://example.com/xAPI/activities`

Tải toàn bộ Activity object được chỉ định.

Giá trị trả về: 200 OK, Content

Tài liệu đơn (PUT | POST | GET | DELETE)

Ví dụ endpoint: `http://example.com/xAPI/activities/profile`

Ghi, sửa, lấy hoặc xóa tài liệu profile cụ thể trong ngữ cảnh Activity chỉ định.

Giá trị trả về (PUT | POST | DELETE): 204 No Content

Giá trị trả về (GET): 200 OK, Profile Content

GET nhiều tài liệu

Ví dụ endpoint: `http://example.com/xAPI/activities/profile`

Tải id của mọi profile entry cho Activity. Nếu chỉ định since, kết quả chỉ gồm entry được ghi/cập nhật sau timestamp chỉ định (không gồm chính thời điểm đó).

Giá trị trả về: 200 OK, List of ids

### 1.6.6 7.6 Agent Profile API

**1.6.6.0.1 Giải thích** Agent Profile API tương tự State API, cho phép lưu cặp key/document tùy ý gắn với Agent.



**1.6.6.0.2 Chi tiết** Ý nghĩa cuộc gọi do tham số `stateId` quyết định. Nếu có tham số này, GET chạy theo tài liệu đơn xác định bởi `profileId`; nếu không, GET trả các id khả dụng.

Agent Profile API cũng gồm phương thức đọc đối tượng đặc biệt kết hợp thông tin Agent với dữ liệu hợp nhất từ dịch vụ ngoài (như directory service).

GET thông tin tổng hợp

Ví dụ endpoint: `http://example.com/xAPI/agents`

Trả về `Person` object đặc biệt cho Agent chỉ định. `Person` tương tự `Agent` object, nhưng mỗi phần tử là mảng giá trị thay vì một giá trị đơn, và cho phép nhiều thuộc tính định danh. Lưu ý tham số vào vẫn là `Agent` object thường với một định danh đơn, không phải mảng. Khái niệm `Person` ở đây không giống FOAF person; ở đây `person` biểu diễn góc nhìn tập trung `person` trên dữ liệu Agent của LRS, còn Agent vẫn chỉ tham chiếu một `persona` (một `person` trong một ngữ cảnh).

#### 1.6.6.0.3 Yêu cầu

- LRS có khả năng trả nhiều thuộc tính định danh trong `Person` nên yêu cầu cấp quyền rõ ràng vì sẽ tăng thông tin định danh liên kết.
- LRS nên từ chối request thiếu quyền bằng 403 `Forbidden`.
- Nếu LRS không có thêm thông tin về Agent, vẫn phải trả `Person` khi truy vấn, nhưng `Person` chỉ chứa thông tin liên quan Agent đã yêu cầu.

Thuộc tính `Person`

#### 1.6.6.0.4 Chi tiết

Tham chiếu: Section 4.1.2.1 Agent.

Giá trị trả về: 200 OK, `Person` Object

#### 1.6.6.0.5 Yêu cầu

- Mọi thuộc tính dạng mảng phải tuân theo cùng định nghĩa với thuộc tính cùng tên trong `Agent` Object.
- Không nên thêm thuộc tính ngoài danh sách vào object này; mỗi thuộc tính chỉ được xuất hiện một lần.

Agent/Profile đơn (PUT | POST | GET | DELETE)

Ví dụ endpoint: `http://example.com/xAPI/agents/profile`

Ghi, sửa, lấy hoặc xóa tài liệu profile cụ thể trong ngữ cảnh Agent chỉ định.

Giá trị trả về (PUT | POST | DELETE): 204 `No Content`

Giá trị trả về (GET): 200 OK, `Profile` Content

GET nhiều Agent/Profile

Ví dụ endpoint: `http://example.com/xAPI/agents/profile`

Tải ID của mọi profile entry cho Agent. Nếu có tham số `since`, kết quả chỉ gồm entry được ghi/cập nhật sau timestamp chỉ định (không gồm chính thời điểm đó).

Giá trị trả về: 200 OK, List of IDs

## 1.6.7 7.7 Tài nguyên About

### Giải thích

Trả thông tin về LRS này (bao gồm các phiên bản xAPI được hỗ trợ).

### Bối cảnh

Tài nguyên này chủ yếu tồn tại để client hỗ trợ nhiều phiên bản xAPI quyết định dùng phiên bản nào khi giao tiếp với LRS. Extensions được giữ để phục vụ các nhu cầu khác có thể xuất hiện.

### Chi tiết

**1.6.7.0.1 GET thông tin** Ví dụ endpoint: `http://example.com/xAPI/about`

Giá trị trả về: 200 OK, tài liệu JSON about đơn.

### Yêu cầu

- Với mỗi major version, LRS phải trả JSON nêu trên kèm thuộc tính `version` chỉ minor/patch mới nhất mà LRS tuân thủ cho major đó.
  - Với bản 1.0.0 của đặc tả này, nghĩa là phải có "1.0.0". Có thể gồm "0.9" hoặc "0.95" (với mục đích yêu cầu này, "0.9" và "0.95" được xem là major version).
- Không được thêm thuộc tính ngoài Extensions vào object này; mỗi thuộc tính chỉ được chỉ định một lần.
- LRS nên cho phép truy cập không xác thực tới tài nguyên này.
- LRS không được từ chối request dựa trên version header trừ khi bắt buộc bởi 6.2 API Versioning.

## 1.6.8 7.8 Yêu cầu Cross-Origin

**1.6.8.0.1 Giải thích** Một mục tiêu của xAPI là cho phép theo dõi liên miền. Dù xAPI muốn hỗ trợ theo dõi từ ứng dụng ngoài trình duyệt, trình duyệt vẫn cần được hỗ trợ. Internet Explorer 8 và 9 không triển khai CORS chuẩn mà dùng cross-domain request API riêng. Chúng chỉ hỗ trợ GET/POST và không cho set HTTP headers, nên không dùng được toàn bộ xAPI như mô tả ở trên.

**1.6.8.0.2 Chi tiết/Yêu cầu** Phần dưới mô tả cú pháp thay thế chỉ nên dùng khi vì hạn chế nêu trên mà không thể dùng cú pháp chuẩn cho một cuộc gọi cụ thể. Cú pháp thay thế này cũng có thể dùng cho GET Statement khi vướng giới hạn độ dài query string.

### Method:

- Mọi xAPI request phát đi phải là POST.
- xAPI method dự kiến phải được chỉ định bằng giá trị tham số query string `method`.
- AP không được chứa gì ngoài query string parameter trong request.

Ví dụ: `http://example.com/xAPI/statements?method=PUT`

### Content:

- Nếu xAPI call có gửi content, AP phải URL-encode content và truyền dưới dạng form parameter tên `content`.

### Header:

- AP có thể chỉ định các header parameter mà đặc tả yêu cầu. Các header này được kỳ vọng xuất hiện như form parameter cùng tên. Áp dụng cho: `Authorization`, `X-Experience-API-Version`, `Content-Type`, `Content-Length`, `If-Match`, `If-None-Match`. Không áp dụng cho `Content-Transfer-Encoding`.

- LRS phải xử lý các form parameter nêu trên như header parameter.
- AP phải gửi các header parameter khác không nằm trong danh sách trên theo cách bình thường.
- AP nên vẫn chỉ định thêm Content-Type header (trong HTTP header) cho các request dạng application/x-www-form-urlencoded.
- Form parameter Content-Type nên chỉ định content type của content nằm trong form parameter content.
- AP nên vẫn chỉ định thêm Content-Length header (trong HTTP header) cho request cho biết tổng độ dài content.
- Form parameter Content-Length nên chỉ định độ dài content trong form parameter content, vì vậy sẽ nhỏ hơn độ dài ghi ở Content-Length header.

#### Query string parameters:

- Query string parameter khác method phải được chỉ định như form parameter cùng tên.
- LRS phải xử lý mọi form parameter (ngoài content và các header parameter nêu trên) như query string parameter.

**Attachments:** Để gửi dữ liệu Attachment cần multipart/mixed request, vì thế cú pháp thay thế này không hỗ trợ gửi Attachment data. Xem 4.1.11 Attachments.

- LRS phải triển khai cú pháp nêu trên.

**Lưu ý:** Internet Explorer phiên bản thấp hơn 10 không hỗ trợ cross-domain request giữa HTTP và HTTPS (tức IE9 trở xuống). Nếu LRS ở HTTPS, client gửi Statement cũng phải ở HTTPS. Nếu LRS ở HTTP, client cũng phải ở HTTP.

Có trường hợp client-side Activity Provider cần hỗ trợ IE8/IE9 và code client được host trên scheme (HTTP/HTTPS) khác với LRS. Khi đó cần proxy để giao tiếp với LRS. Hai cách đơn giản:

1. Cấu hình proxy passthrough cùng scheme từ code client tới LRS.
  2. Host một server-side LRS trung gian có route Statement tới LRS đích, dùng cùng scheme với code client.
- Để hỗ trợ use case này, LRS có thể cung cấp cả HTTP lẫn HTTPS endpoint.
  - LRS và client nên cân nhắc rủi ro bảo mật trước khi quyết định dùng scheme nào.

Xem thêm Appendix G: ví dụ cross-domain request.

## 1.6.9 7.9 Validation

**1.6.9.0.1 Giải thích** Vai trò của LRS trong xAPI là ghi và truy xuất Statement. Khi có đủ thông tin để thực hiện các tác vụ này, LRS được kỳ vọng thực hiện. Validation Statement trong Experience API chỉ ở mức cú pháp, không phán đoán ngữ nghĩa. Việc áp đặt quy tắc đảm bảo ý nghĩa hợp lệ từ tổ hợp định nghĩa Verb, Activity Type và Extensions là trách nhiệm của Activity Provider (nhà cung cấp hoạt động) gửi Statement.

### 1.6.9.0.2 Yêu cầu

- LRS nên áp đặt quy tắc về cấu trúc.
- LRS không nên áp đặt quy tắc về ngữ nghĩa.

## 1.6.10 7.10 HTTP HEAD

Giải thích

LRS phản hồi HEAD request chỉ bằng cách trả metadata qua HTTP headers, không trả nội dung tài liệu.

## Bối cảnh

Client truy cập LRS có thể cần kiểm tra sự tồn tại của Statement cụ thể và xác định ngày sửa tài liệu State/Activity Profile/Agent Profile. Đặc biệt với tài liệu lớn, việc kiểm tra ngày sửa mà không tải toàn bộ tài liệu sẽ hiệu quả hơn.

## Yêu cầu cho LRS

- LRS phải phản hồi mọi HTTP HEAD request giống như phản hồi HTTP GET tương ứng, ngoại trừ:
  - Bỏ message body.
  - Để tránh lãng phí tài nguyên LRS, bỏ header Content-Length.

Ví dụ Statement đơn giản (xuống dòng chỉ để hiển thị).

```
{
  "id": "fd41c918-b88b-4b20-a0a5-a4c32391aaa0",
  "actor": {
    "objectType": "Agent",
    "name": "Project Tin Can API",
    "mbox": "mailto:user@example.com"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/created",
    "display": {
      "en-US": "created"
    }
  },
  "object": {
    "id": "http://example.adlnet.gov/xapi/example/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."
      }
    }
  }
}
```

Ví dụ hoàn thành phổ biến và đơn giản với Verb attempted.

```
{
  "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
  }
}

```

Ví dụ Statement dài giới thiệu phần lớn thuộc tính khả dụng. Ví dụ này cho thấy Statement được LRS trả về có gồm authority và có các thuộc tính do LRS thiết lập.

```

{
  "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/resulttextextensions/minuteslocation": "minuteslocation",
  },
  "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/occurrences/34257",
        "objectType": "Activity"
      }
    ],
  }
}

```

```

        {
            "id": "http://www.example.com/meetings/occurrences/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": "2013-05-18T05:32:34.804Z",
"stored": "2013-05-18T05:32:34.804Z",
"authority": {
    "account": {
        "homePage": "http://cloud.scorm.com/",
        "name": "anonymous"
    },
    "objectType": "Agent"
},
"version": "1.0.0",
"object": {
    "id": "http://www.example.com/meetings/occurrences/34534",
    "definition": {
        "extensions": {
            "http://example.com/profiles/meetings/activitydefinitionextension": {}
        },
        "name": {
            "en-GB": "example meeting",
            "en-US": "example meeting"
        }
    },

```

```

        "description": {
            "en-GB": "An example meeting that happened on a specific occasion",
            "en-US": "An example meeting that happened on a specific occasion",
        },
        "type": "http://adlnet.gov/expapi/activities/meeting",
        "moreInfo": "http://virtualmeeting.example.com/345256"
    },
    "objectType": "Activity"
}

```

Object của Statement có thể là Activity, Group hoặc Statement. Phụ lục này đưa ra ví dụ cho từng loại.

#### 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"
}

```

#### Agent

```

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

```

#### Group

Ví dụ này thể hiện một identified group có thành viên.

```

{
    "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"
    }
]
}

```

## Statement

Ví dụ này thể hiện Sub-Statement có object là StatementRef.

```

{
    "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"
    }
}

```

## 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"
    ]
}

```

## 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"
        }
    }
]
}

```

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"
    ]
}

```

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 ex
    ]
}

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"
            }
        }
    ]
}

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"
    }
}
]
}

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"
            }
        }
    ]
}

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"
            }
        }
    ]
}

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[:]"
    ]
}

```

Trong ví dụ này, giá trị đúng tối thiểu là 4 và không có giá trị tối đa. Các số như 5, 6, 976 đều là đáp án đúng.

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)"
  ]
}

```

## Bối cảnh

Đặc tả này là phiên bản 1.0.0 và cố gắng để nhà triển khai không phải quan tâm phiên bản cũ. Tuy nhiên các phiên bản trước đã có mức độ phổ biến đáng kể. Quy tắc chuyển đổi dữ liệu này nhằm giúp tiếp tục tận dụng dữ liệu ghi theo phiên bản cũ, đồng thời để nhà triển khai mới cũng có thể dùng dữ liệu đó theo cách tương tự.

## Chi tiết

Chuyển Statement tạo từ phiên bản 0.9

Hệ thống 1.0.0 chuyển đổi Statement từ 0.9 phải làm theo các bước sau:

- Không chuyển đổi nếu Statement là voided, hoặc có verb/activity type/property không thuộc đặc tả 0.9.
- Gắn prefix `http://adlnet.gov/expapi/verbs/` vào Verb.
- Với Activity ID không phải absolute IRI, gắn prefix tag: `adlnet.gov,2013:expapi:0.9:activities:`.
- Với extension key không phải full absolute IRI, gắn prefix tag: `adlnet.gov,2013:expapi:0.9:activities:`.
- Gắn prefix `http://adlnet.gov/expapi/activities/` vào Activity Type.
- Với mọi Agent (actor):
  - Tìm inverse functional identifier theo thứ tự "mbox, mbox\_sha1sum, openId, account". Giữ cái đầu tiên tìm được, bỏ phần còn lại.
  - Với inverse functional identifier đã giữ, lấy phần tử đầu tiên của mảng làm giá trị, bỏ các phần tử còn lại.
  - Nếu có name, đặt bằng phần tử đầu tiên của mảng name, bỏ phần còn lại.
  - Xóa toàn bộ thuộc tính còn lại.
- Nếu có thuộc tính voided trong Statement thì xóa đi. Nếu voided = true thì không chuyển đổi.
- Thêm "version": "1.0.0".
- Nếu authority chưa được thiết lập sẵn, đặt authority là Agent định danh bằng account có homePage là trang của hệ thống chạy chuyển đổi và accountName = "unknown".
- Nếu có thuộc tính statement trong context thì xóa khỏi Statement.
- Giữ nguyên các thuộc tính khác, gồm cả stored (nhưng stored sẽ được cập nhật khi Statement được chuyển sang hệ thống khác).

Chuyển Statement tạo từ phiên bản 0.95

Hệ thống 1.0.0 chuyển đổi Statement từ 0.95 phải làm theo các bước sau:

- Không chuyển đổi Statement đã void.
- Nếu có thuộc tính voided, xóa khỏi Statement. Nếu voided = true thì không chuyển đổi Statement.
- Thêm "version": "1.0.0".
- Nếu authority chưa được thiết lập sẵn, cấp authority cho Agent định danh bằng account có homePage tương ứng hệ thống chạy chuyển đổi và accountName = "unknown".
- Nếu thuộc tính statement trong context được đặt bằng giá trị khác StatementRef, xóa thuộc tính này khỏi Statement.

- Giữ nguyên các thuộc tính khác, gồm cả stored (nhưng stored sẽ được cập nhật khi Statement được chuyển sang hệ thống khác).

Ví dụ

#### A 0.9 Statement

```
{
  "id": "d1eec41f-1e93-4ed6-acbf-5c4bd0c24269",
  "actor": {
    "objectType": "Person",
    "name": [
      "Joe Schmoe",
      "Joseph Schmoseph"
    ],
    "mbox": [
      "mailto:joe@example.com"
    ],
    "openid": [
      "http://openid.com/joe-schmoe"
    ]
  },
  "verb": "completed",
  "inProgress": false,
  "object": {
    "objectType": "Activity",
    "id": "http://www.example.com/activities/001",
    "definition": {
      "name": {
        "en-US": "Example Activity"
      },
      "type": "course"
    }
  },
  "result": {
    "completion": true
  },
  "context": {
    "instructor": {
      "objectType": "Person",
      "lastName": [
        "Dad"
      ],
      "firstName": [
        "Joe's"
      ],
      "mbox": [
        "mailto:joesdad@example.com"
      ]
    },
  },
```



```

    "contextActivities": {
      "parent": {
        "objectType": "Activity",
        "id": "non-absolute-activity-id",
        "definition": {
          "name": {
            "en-US": "Another Activity"
          }
        }
      }
    },
    "timestamp": "2012-06-01T19:09:13.245Z",
    "stored": "2012-06-29T15:41:39.165Z"
  }
}

```

Converted to 1.0.0:

```

{
  "version": "1.0.0",
  "id": "d1eec41f-1e93-4ed6-acbf-5c4bd0c24269",
  "actor": {
    "objectType": "Agent",
    "name": "Joe Schmoe",
    "mbox": "mailto:joe@example.com"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/completed",
    "display": {
      "en-US": "completed"
    }
  },
  "object": {
    "objectType": "Activity",
    "id": "http://www.example.com/activities/001",
    "definition": {
      "name": {
        "en-US": "Example Activity"
      },
      "type": "http://adlnet.gov/expapi/activities/course"
    }
  },
  "result": {
    "completion": true
  },
  "context": {
    "instructor": {
      "objectType": "Agent",
      "mbox": "mailto:joesdad@example.com"
    }
  },
}

```

```

    "contextActivities": {
      "parent": [
        {
          "objectType": "Activity",
          "id": "tag:adlnet.gov,2013:expapi:0.9:activities:non-absolut
          "definition": {
            "name": {
              "en-US": "Another Activity"
            }
          }
        }
      ]
    },
    },
    "timestamp": "2012-06-01T19:09:13.245Z",
    "stored": "2012-06-29T15:41:39.165Z",
    "authority": {
      "objectType": "Agent",
      "account": {
        "homePage": "http://www.example.com",
        "name": "unknown"
      }
    }
  }
}

```

Danh sách Statement gốc cần được ký. Trong ví dụ này, dòng mới bao gồm 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

```

```

    }
  },
  "timestamp": "2013-04-01T12:00:00Z"
}

```

Ví dụ private key của chứng chỉ X.509 dùng để ký:

```

-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAABgQDjxvZXF30WL4oKjZYXgR0ZyaX+u3y6+JqTqiNkFa/VTnet6Ly2
OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkWl3j06j5iNn4obcCVWTL9yXNvJH+Ko+xu4Y
l/ySPRRIPyTjtHdG0M2XzIlmmLqm+CAS+KCbJeH4tf543kIWC5pC5p3cVQIDAQAB
AoGA0ejdvGq2XKuddu1kWXl0Aphn4YmdPpPyCNTaxplU6PBMYRjY0aNgLQE6bO2p
/HJiU4Y4PkgzkEgCu0xf/mOq5DnSkX32ICoQS6jChABAe20ErPfm5t8h9YKsTfn9
40lAouuwD9ePRteizd4YvHtiMMwmh5PtUoCbqLefawNApAECQQDlmdBW3zL0okUx
2pc4tttn2qArCG4CsEZMLlGRDd3FwPWJz3ZPNEEGZWXGSpA9F1QTZ6JYXIfejRo
UuvRMWeBAkEA7WvzDBNcv4N+xeUKvH8ILti/BM58LraTtqJlZjQSovek0srxtmDg
5of+xrxN6IM4p7yvQa+7YOUOukrVXjG+1QJBAI2mBrjzxgm9xTa5odn97JD7UMFA
/WHjlMe/Nx/35V52qaavlsZbluw+TvKMcqApYj5G2SUPsNudHLDGkmd2nQECQFfc
lBRK8g7ZncekbGW3aRLVGVOxClNLLTzwOlambKOUm8V6XxsMHQ6TE2D+fKJoNUY1
2HGpk+FWwy2D1hRGuoUCQAXfaLSxtaWdPtLzTPVueF7ZikQDsVg+vtTFgpuHloR2
6EVclRbHHZm32yvGDY8IkcoMfJQqLONDdLfs/05yoNU=
-----END RSA PRIVATE KEY-----

```

Ví dụ public X.509 certificate:

```

-----BEGIN CERTIFICATE-----
MIIDATCCAmqgAwIBAgIJAMB1csNuA6+kMA0GCSqGSIb3DQEBAQUAMHExCzAJBgNV
BAYTA1VTMRIwEAYDVQQIEwluZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUgQ29t
cGFueTEQMA4GA1UEAxMHRXhbbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhbbXBsZUBl
eGftcGx1LmNvbTAeFw0xMzA0MDQxNTI4MzBaFw0xNDA0MDQxNTI4MzBaMIGWMSw
CQYDVQQGEwJVUzESMBAGA1UECBMJVGvubmVzc2VlMREwDwYDVQQHEwhGcmFua2xp
bjEYMBYGA1UEChMPRXhbbXBsZSBDb21wYW55MRAwDgYDVQQLEwdFeGftcGx1MRAw
DgYDVQQDEwdFeGftcGx1MSIwIAYJKoZIhvcNAQkBFhNleGftcGx1QGV4YW1wbGUu
Y29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDjxvZXF30WL4oKjZYXgR0Z
yaX+u3y6+JqTqiNkFa/VTnet6Ly2OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkWl3j06j
5iNn4obcCVWTL9yXNvJH+Ko+xu4Yl/ySPRRIPyTjtHdG0M2XzIlmmLqm+CAS+KCb
JeH4tf543kIWC5pC5p3cVQIDAQABo3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIB
DQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBdZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUVs3v
5afEdOeoYeVajAQE4v0WS1QwHwYDVR0jBBgwFoAUyVlc3yvra4EBz20I4BF39IAi
xBkWdQYJKoZIhvcNAQEFBQADgYEAgS/FF5D0Hnj44rvT6kgn3kJAvv2lj/fyztK
IrYS33ljXGn6gGyA4qtbXA23PrO4uc/wYCSdICDpPobh62xTCd9qObKhgwWoi05P
SBLqUu3mwfAe15LJBjBqPVZ4K0kppePBU8m6aIZoH57L/9t4OoaL8yKs/qjKFeI1
OFWZxvA=
-----END CERTIFICATE-----

```

Ví dụ CA certificate:

```

-----BEGIN CERTIFICATE-----
MIIDNzCCAqCgAwIBAgIJAMB1csNuA6+jMA0GCSqGSIb3DQEBAQUAMHExCzAJBgNV
BAYTA1VTMRIwEAYDVQQIEwluZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUgQ29t
cGFueTEQMA4GA1UEAxMHRXhbbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhbbXBsZUBl

```

```
eGFtcGx1LmNvbTAeFw0xMzA0MDQxNTI1NTNaFw0yMzA0MDIxNTI1NTNaMHExCzAJ
BgNVBAYTA1VTMRIwEAYDVQQIEWlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUg
Q29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIB3DQEJARYTZXhhbXBs
ZUBleGFtcGx1LmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA1sBnBWPZ
0f7WJUFTJy5+01S1S5Z6DDD6Uye9vK9AycgV5B3+WC8HC5u5h91MujAC1ARPVUOt
svPRs45qKNFIgIGRXKPAwZjAwEI2sCJRSKV47i6B8bDv4WkuGvQaveZGI0qlmN5R
1Eim2gUITrj1hgC9rQavjlnFKDY2rlXGukCAwEAAaOB1jCB0zAdBgNVHQ4EFgQU
yVic3yvra4EBz20I4BF39IAixBkwgaMGA1UdIwSBmzCBmIAUyVic3yvra4EBz20I
4BF39IAixBmhdaRzMHExCzAJBgNVBAYTA1VTMRIwEAYDVQQIEWlUZW5uZXNzZWUx
GDAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAG
CSqGSIB3DQEJARYTZXhhbXBsZUBleGFtcGx1LmNvbYIJAMB1csNuA6+jMAwGA1Ud
EwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEADhwTebGk735yKhm8DqCxxvNnEZ0Nx
sYEYOjgRGlyXTlW5pE691fSH5AZ+T6fpwpZcWY5QYkoN6DnwjOxGkSfQC3/yGmcU
DKBPwiz5O2s9C+fElkUEnrX2Xea4agVngMzR8DQ6oOauLWqehDB+g2ENWRLovGs+
ma5/Ycs0GTyrECY=
-----END CERTIFICATE-----
```

JWS header. Lưu ý ngoài thuật toán, còn có certificate chain cho chứng chỉ ký.

```
{
  "alg": "RS256",
  "x5c": [
    "MIIDATCCAmqgAwIBAgIJAMB1csNuA6+kMA0GCSqGSIB3DQEBBQUAMHExCzAJBgNVBAYTA1V
    "MIIDNzCCAqCgAwIBAgIJAMB1csNuA6+jMA0GCSqGSIB3DQEBBQUAMHExCzAJBgNVBAYTA1V
  ]
}
```

JWS signature

```
ew0KICAgICJhbGciOiAiA1UyMyNTYiLA0KICAgICJ4NWMiOiBbDQogICAgICAgICJNSU1EQVRDQ0FtcWdl
```

Statement đã ký:

```
{
  "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
    }
}
},
"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
    }
]
}

```

**Ghi chú:** Nếu chữ ký đính kèm không hiển thị, hãy kiểm tra format của attachment message trong attachments.

### 1.6.11 Endpoint bắt buộc

### 1.6.12 OAuth Endpoint

7.8 Cross Origin Requests mô tả cú pháp thay thế khi không thể dùng cú pháp chuẩn do hạn chế trình duyệt hoặc độ dài query string. Phụ lục này đưa ví dụ PUT Statement theo định dạng thay thế đó.

Cú pháp chuẩn:

URL: `http://example.com/xAPI/statements`

Method: PUT

Query String Parameters:

`statementId=c70c2b85-c294-464f-baca-cebd4fb9b348`

Request Headers:

`Accept: */*`

`Accept-Encoding: gzip, deflate, sdch`

`Accept-Language: en-US, en; q=0.8`

`Authorization: Basic VGVzdFVzZXI6cGFzc3dvcmQ=`

`Content-Type: application/json`

`X-Experience-API-Version: 1.0.3`

`Content-Length: 351`

Content:

`{"id": "c70c2b85-c294-464f-baca-cebd4fb9b348", "timestamp": "2014-12-29T12:09:37.4"`

Request dùng cú pháp thay thế:

URL: http://example.com/xAPI/statements?method=PUT&statementId=c70c2b85-c294-464f-baca-cebd4fb9b348  
Method: POST

Request Headers:

Accept: \*/\*  
Accept-Encoding: gzip, deflate, sdch  
Accept-Language: en-US,en;q=0.8  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 745

Content (đã thêm xuống dòng và chưa URL-encode để dễ đọc):

statementId=c70c2b85-c294-464f-baca-cebd4fb9b348  
&Authorization=Basic VGVzdFVzZXI6cGFzc3dvcmQ=  
&X-Experience-API-Version=1.0.3  
&Content-Type=application/json  
&Content-Length=351  
&content={"id":"c70c2b85-c294-464f-baca-cebd4fb9b348","timestamp":"2014-12-2