xAPI Statements

2.1 Purpose

Statements are the evidence for any sort of experience or event which is to be tracked in xAPI. While Statements follow a machine readable JSON format, they can also easily be described using natural language. This can be extremely useful for the design process. Statements are meant to be aggregated and analyzed to provide larger meaning for the overall experience than just the sum of its parts.

2.2 Formatting Requirements

Details

All of the properties used in Statements are restricted to certain data types. For clarity, key requirements are documented here, emphasizing where xAPI components have a responsibility to act in certain ways to be considered conformant to this specification.

Requirements

- Statements and other objects SHOULD NOT include properties with a value of an empty object.
- A Statement MUST use each property no more than one time.
- A Statement MUST use "actor", "verb", and "object".
- A Statement MAY use its properties in any order.
- The LRS MUST NOT return a different serialization of any properties except those listed as exceptions.

Learning Record Provider Requirements

The following requirements reiterate especially important requirements already included elsewhere, to emphasize, clarify, and provide implementation guidance. Some types of validation, such as complete IRI validation and validation that properties appear only once, are extremely difficult, so much of the burden for ensuring data portability is on the Learning Record Provider.

- Values requiring IRIs MUST be sent with valid IRIs.
- Keys of language maps MUST be sent with valid RFC 5646 language tags, for similar reasons.
- A library SHOULD be used to construct IRIs, as opposed to string concatenation.

- Values SHOULD be considered to be case sensitive unless specified otherwise.
- Lowercase SHOULD be used to send case insensitive data.
- Additional properties SHOULD* NOT be added to Statements unless explicitly allowed by this specification.
- A property SHOULD not occur multiple times in an object. If properties are used multiple times within an object, the behavior of the LRS is undefined; it is expected that most LRSs will use existing JSON parsing functionality of whichever code languages they use.

Note: The LRS is recommended to reject Statements containing additional properties. Additional properties in Statements would mean that the Statement would not be interoperable with all LRSs.

LRS Requirements

- The LRS MUST reject Statements
  - with any null values (except inside extensions).
  - where the wrong data type is used, for example:
    - with strings where numbers are required, even if those strings contain numbers, or
    - with strings where booleans are required, even if those strings contain booleans.
  - with any non-format-following key or value, including the empty string, where a string with a particular format (such as mailto IRI, UUID, or IRI) is required.
  - where the case of a key does not match the case specified in this specification.
  - where the case of a value restricted to enumerated values does not match an enumerated value given in this specification exactly.
  - where a key or value is not allowed by this specification.
  - containing IRL or IRI values without a scheme.
- The LRS MUST at least validate that the sequence of token lengths for language map keys matches the RFC 5646 standard.
- The LRS MUST process and store numbers with at least the precision of IEEE 754 32-bit floating point numbers.
- The LRS MUST validate parameter values to the same standards required for values of the same types in Statements. **Note:** string parameter values are not quoted as they are in JSON.

- The LRS SHOULD treat all values as case sensitive unless specified otherwise.
- The LRS MAY use best-effort validation for IRL and IRI formats to satisfy the non-format-following rejection requirement.
- The LRS MAY use best-effort validation for language map keys to satisfy the non-format-following rejection requirement.
- Additional properties SHOULD* NOT be added to Statements and other objects unless explicitly allowed by this specification and the LRS SHOULD* reject Statements containing such additional properties.

2.3 Statement Lifecycle

Statements are information about a tracked learning experience. Typically, the information represented in the Statement has already happened. Thus, the natural language used in "display" or in the human-readable portion of the Verb id will usually use the past tense.

Statements are expected to be permanent. The only way to undo a Statement within this specification is to void it. Voiding does not destroy a Statement, rather indicates the evidence in the Statement is to be disregarded.

2.3.1 Statement Immutability

Statements are immutable (they cannot be changed). The following are exceptions or areas not covered by this rule:

- Potential or required assignments of properties during LRS processing ("id", "authority", "stored", "timestamp", "version").
- Activity Definitions referenced by a Statement. The content of Activity Definitions that are referenced in Statements is not considered part of the Statement itself. This means a deep serialization of a Statement into JSON will change if the referenced Activity Definition changes (see the Statement Resource's "format" parameter for details).
- Verbs referenced by a Statement. The "display" property of the Verb is not considered part of the Statement itself (see the Statement Resource's "format" parameter for details).
- Serialization of Timestamp data. This is not considered part of the immutable Statement itself. For example, the "timestamp" and "stored" properties of a Statement can be returned in a different timezone to the one with which they were stored so long as the point in time referenced is not affected. See 2.4.7 Timestamp and 2.4.8 Stored for details.

- Serialization of un-ordered lists. The list of Agents in a Group is not considered to be an ordered list. Thus, the LRS can return this list of Agents in any order. See Groups.
- Attachments. These are not part of Statements and an LRS will return Statements without Attachments when a Client requests them (see the Statement Resource's "attachments" parameter for details).
- Case sensitivity. Some properties are case insensitive and changes in case therefore do not affect immutability. For example, the domain portion an e-mail address is case insensitive. It is recommended to use lowercase for any case insensitive text.

The following explicitly are **not** exceptions and **are** covered by this rule:

- Result Duration. Due to variable lengths of months, years and even minutes and the flexible nature of the "timestamp" property as representing either the start, middle or end of the experience, it is not possible for an LRS to accurately deserialize the Result Duration and convert between units of time. For this reason, the Result Duration is considered a string for purposes of Statement comparison.

Statement Comparision Requirements

There are a number of scenarios outlined in this specification which require Statements to be compared to see if they match. In these scenarios, the following rules apply:

- Differences which could have been caused by exceptions to Statement immutability MUST be ignored.
- Differences relating to a different serialization of any properties not listed as exceptions MUST not be ignored.

2.3.2 Voiding

Rationale

The certainty that an LRS has an accurate and complete collection of data is guaranteed by the fact that Statements cannot be logically changed or deleted. This immutability of Statements is a key factor in enabling the distributed nature of Experience API.

However, not all Statements are perpetually valid once they have been issued. Mistakes or other factors could dictate that a previously made Statement is marked as invalid. This is called "voiding a Statement" and the reserved

Verb `http://adlnet.gov/expapi/verbs/voided` is used for this purpose. Any Statement that voids another cannot itself be voided.

Requirements

- When issuing a Statement that voids another, the Object of that voiding Statement MUST have the "objectType" property set to `StatementRef`.
- When issuing a Statement that voids another, the Object of that voiding Statement MUST specify the id of the Statement-to-be-voided by its "id" property.
- An LRS MUST consider a Statement it contains voided if and only if the Statement is not itself a voiding Statement and the LRS also contains a voiding Statement referring to the first Statement.
- Upon receiving a Statement that voids another, the LRS SHOULD reject the entire request which includes the voiding Statement with `403 Forbidden` if the request is not from a source authorized to void Statements.
- Upon receiving a Statement that voids another, the LRS SHOULD NOT* reject the request on the grounds of the Object of that voiding Statement not being present.
- Upon receiving a Statement that voids another, the LRS MAY roll back any changes to Activity or Agent definitions which were introduced by the Statement that was just voided.
- A Learning Record Provider that wants to "unvoid" a previously voided Statement SHOULD issue that Statement again under a new id.

Note: See "Statement References" in When the "Object" is a Statement for details about making references to other Statements. To see how voided statements behave when queried, See StatementRef in Part 3).

Example

This example Statement voids a previous Statement which it identifies with the Statement 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"
        }
}
```

2.4 Statement Properties

Details

The details of each property of a Statement are described in the table below.

Property	Type	Description	Required
id	UUID	UUID assigned by LRS if not set by the Learning Record Provider.	Recommended
actor	Object	Whom the Statement is about, as an Agent or Group Object.	Required
verb	Object	Action taken by the Actor.	Required
object	Object	Activity, Agent, or another Statement that is the Object of the Statement.	Required
result	Object	Result Object, further details representing a measured outcome.	Optional
context	Object	Context that gives the Statement more meaning. Examples: a team the Actor is working with, altitude at which a scenario was attempted in a flight simulator.	Optional
timestamp	Timestamp	Timestamp of when the events described within this Statement occurred. Set by the LRS if not provided.	Optional
stored	Timestamp	Timestamp of when this Statement was recorded. Set by LRS.	Set by LRS
authority	Object	Agent or Group who is asserting this Statement is true. Verified by the LRS based on authentication. Set by LRS if not provided or if a strong trust relationship between the Learning	Optional

		Record Provider and LRS has not been established.	
version	Version	The Statement’s associated xAPI version, formatted according to Semantic Versioning 1.0.0.	Not Recommended
attachments	Ordered array of Attachment Objects	Headers for Attachments to the Statement	Optional

Example

An example of the simplest possible Statement using all properties that MUST or SHOULD be used. It is recommended to also populate optional properties where relevant. When this Statement is returned from the LRS it will include some additional properties added by the LRS.

```
{
    "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"
    }
}
```

See Appendix A: Example Statements for more examples.

2.4.1 ID

Description

A UUID (all versions of variant 2 in RFC 4122 are valid, and the UUID MUST be in standard string form).

Requirements

- Statement ids MUST be generated by the LRS if a Statement is received without an id.

- Statement ids SHOULD be generated by the Learning Record Provider.

2.4.2 Actor

Description

The Actor defines who performed the action. The Actor of a Statement can be an Agent or a Group.

2.4.2.1 When the Actor objectType is Agent

Description

An Agent (an individual) is a persona or system.

Details

- An Agent MUST be identified by one (1) of the four types of Inverse Functional Identifiers (see 2.4.2.3 Inverse Functional Identifier);
- An Agent MUST NOT include more than one (1) Inverse Functional Identifier;
- An Agent SHOULD NOT use Inverse Functional Identifiers that are also used as a Group identifier.

The table below lists the properties of Agent Objects.

Property	Type	Description	Required
objectType	string	`Agent`. This property is optional except when the Agent is used as a Statement's object.	Optional
name	String	Full name of the Agent.	Optional
see 2.4.2.3 Inverse Functional Identifier		An Inverse Functional Identifier unique to the Agent.	Required

2.4.2.2 When the Actor ObjectType is Group

Description

A Group represents a collection of Agents and can be used in most of the same situations an Agent can be used. There are two types of Groups: Anonymous Groups and Identified Groups.

Details

An Anonymous Group is used to describe a cluster of people where there is no ready identifier for this cluster, e.g. an ad hoc team.

The table below lists all properties of an Anonymous Group.

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

An Identified Group is used to uniquely identify a cluster of Agents.

The table below lists all properties of an Identified Group.

Property	Type	Description	Required
objectType	String	`Group`.	Required
name	String	Name of the Group.	Optional
member	Array of Agent Objects	The members of this Group. This is an unordered list.	Optional
see 2.4.2.3 Inverse Functional Identifier		An Inverse Functional Identifier unique to the Group.	Required

Requirements

- A Learning Record Consumer MUST consider each Anonymous Group distinct even if it has an identical set of members.
- Learning Record Providers SHOULD use an Identified Group when they wish to issue multiple Statements, aggregate data or store and retrieve documents relating to a group.
- A Learning Record Provider MAY include a complete or partial list of Agents in the "member" property of a given Anonymous or Identified Group.
- An LRS returning a Statement MAY return the list of Group members in any order.

Requirements for Anonymous Groups

- An Anonymous Group MUST include a "member" property listing constituent Agents.
- An Anonymous Group MUST NOT contain Group Objects in the "member" identifiers.

Requirements for Identified Groups

- An Identified Group MUST include exactly one (1) Inverse Functional Identifier.
- An Identified Group MUST NOT contain Group Objects in the "member" property.
- An Identified Group SHOULD NOT use Inverse Functional Identifiers that are also used as Agent identifiers.
- An Identified Group MAY include a "member" property listing constituent Agents.

2.4.2.3 Inverse Functional Identifier

Description

An Inverse Functional Identifier (IFI) is a value of an Agent or Identified Group that is guaranteed to only ever refer to that Agent or Identified Group.

Rationale

Agents and Groups need to be uniquely identifiable in order for data to be stored and retrieved against them. In an xAPI Statement this is accomplished using Inverse Functional Identifiers which are loosely inspired by the widely accepted FOAF principle (see: Friend Of A Friend).

Details

The table below lists all possible Inverse Functional Identifier properties.

Property	Type	Description
mbox	mailto IRI	The required format is "mailto:email address". Only email addresses that have only ever been and will ever be assigned to this Agent, but no others, SHOULD be used for this property and mbox_sha1sum.
mbox_sha1sum	String	The hex-encoded SHA1 hash of a mailto IRI (i.e. the value of an mbox property). An LRS MAY include Agents with a matching hash when a request is based on an mbox.
openid	URI	An openID that uniquely identifies the Agent.

account	Object	A user account on an existing system e.g. an LMS or intranet.

Client Requirements

- The domain portions of email addresses are case insensitive. Clients SHOULD use lowercase for the domain portion of the email address when calculating the SHA1 hash for the "mbox_sha1sum" property.

2.4.2.4 Account Object

Description

A user account on an existing system, such as a private system (LMS or intranet) or a public system (social networking site).

Details

- If the system that provides the account Object uses OpenID, the Learning Record Provider SHOULD use the openid property instead of an account Object.
- If the Learning Record Provider is concerned about revealing personally identifiable information about an Agent or Group, it SHOULD use an opaque account name (for example an account number) to identify all Statements about a person while maintaining anonymity.

The table below lists all properties of Account Objects.

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

Example

This example shows an Agent identified by an opaque account:

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

2.4.3 Verb

Description

The Verb defines the action between an Actor and an Activity.

Rationale

The Verb in an xAPI Statement describes the action performed during the learning experience. The xAPI does not specify any particular Verbs. (With one exception, namely the reserved Verb http://adlnet.gov/expapi/verbs/voided). Instead, it defines how to create Verbs so that communities of practice can establish Verbs meaningful to their members and make them available for use by anyone. A predefined list of Verbs would be limited by definition and might not be able to effectively capture all possible future learning experiences.

Details

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

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

Verb Id Requirements

- A system reading a Statement MUST use the Verb IRI to infer meaning.

- The IRI contained in an id SHOULD contain a human-readable portion which SHOULD provide meaning enough for a person reviewing the raw statement to disambiguate the Verb from other similar (in syntax) Verbs.
- A single Verb IRI MUST NOT be used to refer to multiple meanings.

Verb Display Learning Record Provider Requirements

- The "display" property SHOULD be used by all Statements.
- The "display" property MUST be used to illustrate the meaning which is already determined by the Verb IRI.

Verb Display LRS Requirements

The requirements below relate to the "display" property as returned by the LRS via the API.

- When queried for Statements with a Format of `exact`, the LRS MUST return the "display" property exactly as included (or omitted) within the Statement.
- When queried for Statements with a Format of `ids`, the LRS SHOULD* NOT include the "display" property.
- When queried for Statements with a Format of `canonical`, the LRS SHOULD* return a canonical Display for that Verb.
- The LRS may determine its canonical Display based on the Verb's "display" property included within Statements it receives, the "name" property included in the metadata as described in 3.2 Hosted Metadata, or the Verb's Display as defined in some other location.

Verb Display Learning Record Consumer Requirements

The requirements below relate to the display property as displayed to a user by a Learning Record Consumer.

- The "display" property MUST NOT be used to alter the meaning of a Verb.
- A Learning Record Consumer MUST NOT use the "display" property to infer any meaning from the Statement.
- A Learning Record Consumer MUST NOT use the "display" property for any purpose other than to display to a human. Using the "display" property for aggregation or categorization of Statements is an example of violating this requirement.

- A Learning Record Consumer displaying a Statement's Verb in a user interface MAY choose to render the Verb's "display" property included within the Statement, the "name" property included in the metadata as described in 3.2 Hosted Metadata, or the Verb's Display as defined in some other location.
- Learning Record Consumers displaying a Statement's Verb MUST NOT display a word that differs from the meaning of the Verb but MAY alter the wording and tense displayed for the purposes of human-readability.

Example

This example shows a Verb with the recommended properties set and using US English and Spanish languages.

```
{
    "id":"http://example.com/xapi/verbs#defenestrated",
    "display":{
        "en-US":"defenestrated",
        "es" : "defenestrado"
    }
}
```

The Verb in the example above is included for illustrative purposes only. This is not intended to imply that a Verb with this meaning has been defined with this id. This applies to all example verbs given in this specification document, with the exception of the reserved Verb http://adlnet.gov/expapi/verbs/voided).

2.4.3.1 Use in Language and Semantics of Verbs

Details

Semantics

The IRI represented by the Verb id identifies the particular semantics of a word, not the word itself.

For example, the English word "fired" could mean different things depending on context, such as "fired(a weapon)", "fired(a kiln)", or "fired(an employee)". In this case, an IRI identifies one of these specific meanings.

The "display" property has some flexibility in tense. While the human-readable portion of the Verb id will usually use the past tense, if conjugating verbs to another tense within the "display" property makes the most sense for the Statement as a whole, it is allowed.

Language

A Verb in the Experience API is an IRI, and denotes a specific meaning not tied to any particular language.

For example, a particular Verb IRI such as http://example.org/firearms#fire might denote the action of firing a gun, or the Verb IRI http://example.com/فعل/خواندن might denote the action of reading a book.

2.4.4 Object

Description

The Object defines the thing that was acted on. The Object of a Statement can be an Activity, Agent/Group, SubStatement, or Statement Reference.

Some examples:

- The Object is an Activity: "Jeff wrote an essay about hiking."
- The Object is an Agent: "Nellie interviewed Jeff."
- The Object is a SubStatement or Statement Reference (different implementations, but similar when human-read): "Nellie commented on 'Jeff wrote an essay about hiking.'"

Details

Objects which are provided as a value for this property SHOULD include an "objectType" property. If not specified, the objectType is assumed to be `Activity`. Other valid values are: `Agent`, `Group`, `SubStatement` or `StatementRef`. The properties of an Object change according to the objectType.

2.4.4.1 When the ObjectType is Activity

Details

A Statement can represent an Activity as the Object of the Statement. The following table lists the Object properties in this case.

Property	Type	Description	Required
objectType	String	MUST be `Activity` when present	Optional
id	IRI	An identifier for a single unique Activity	Required
definition	Object	Metadata, See below	Optional

If it were possible to use the same id for two different Activities, the validity of Statements about these Activities could be questioned. This means an LRS can't ever treat (references to) the same Activity id as belonging to two different Activities, even if it thinks this was intended. Namely, when a conflict with another system occurs, it’s not possible to determine which Activity is intended.

Activity Definition

The table below lists the properties of the Activity Definition Object:

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

Note: IRI fragments (sometimes called relative IRLs) are not valid IRIs. As with Verbs, it is recommended that those implementing xAPI look for and use established, widely adopted, Activity types.

Activity id Requirements

- An Activity id MUST be unique.
- An Activity id MUST always reference the same Activity.
- An Activity id SHOULD use a domain that the creator is authorized to use for this purpose.
- An Activity id SHOULD be created according to a scheme that makes sure all Activity ids within that domain remain unique.
- An Activity id MAY point to metadata or the IRL for the Activity.

LRS Requirements

- An LRS MUST NOT take action in the event it perceives an Activity id is being used by multiple authors and/or organizations.
- An LRS MUST NOT treat references to the same Activity id as references to different Activities.
- Upon receiving a Statement with an Activity Definition that differs from the one stored, an LRS SHOULD decide whether it considers the Learning Record Provider to have the authority to change the definition and SHOULD update the stored Activity Definition accordingly if that decision is positive.
- An LRS MAY make small corrections to its canonical definition for the Activity when receiving a new definition e.g. to fix a spelling.
- An LRS SHOULD NOT make significant changes to its canonical definition for the Activity based on an updated definition e.g. changes to correct responses.

Learning Record Provider Requirements

- A Learning Record Provider MUST ensure that Activity ids are not used across multiple Activities.
- A Learning Record Provider MUST only generate states or Statements against a certain Activity id that are compatible and consistent with states or Statements previously stored against the same Activity id.
- A Learning Record Provider MUST NOT allow new versions (i.e. revisions or other platforms) of the Activity to break compatibility.

Interaction Activities

Rationale

Traditional e-learning has included structures for interactions or assessments. As a way to allow these practices and structures to extend Experience API's utility, this specification includes built-in definitions for interactions, which borrows from the SCORM 2004 4th Edition Data Model. These definitions are intended to provide a simple and familiar utility for recording interaction data. Since 1.0.3, direct references to the SCORM data model have started to be removed, and any associated requirements included directly in this document.

These interaction definitions are simple to use, and consequently limited. It is expected that Communities of Practice requiring richer interactions definitions will do so through the use of Activity types and Activity Definition Extensions.

Details

The table below lists the properties for Interaction Activities.

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

Interaction Types

The table below describes the kinds of interactions represented by each of the interactionTypes. These types of interactions were originally based on the types of interactions allowed for "cmi.interactions.n.type" in the SCORM 2004 4th Edition Run-Time Environment. See Appendix C for examples definitions for each interaction type.

interactionType	Description
true-false	An interaction with two possible responses: `true` or `false`.
choice	An interaction with a number of possible choices from which the learner can select. This includes interactions in which the learner can select only one answer from the list and those where the learner can select multiple items.
fill-in	An interaction which requires the learner to supply a short response in the form of one or more strings of characters. Typically, the correct response consists of part of a word, one word or a few words. "Short" means that the correct responses pattern and learner response strings will normally be 250 characters or less.
long-fill-in	An interaction which requires the learner to supply a response in the form of a long string of characters. "Long" means that the correct

	responses pattern and learner response strings will normally be more than 250 characters.
matching	An interaction where the learner is asked to match items in one set (the source set) to items in another set (the target set). Items do not have to pair off exactly and it is possible for multiple or zero source items to be matched to a given target and vice versa.
performance	An interaction that requires the learner to perform a task that requires multiple steps.
sequencing	An interaction where the learner is asked to order items in a set.
likert	An interaction which asks the learner to select from a discrete set of choices on a scale
numeric	Any interaction which requires a numeric response from the learner.
other	Another type of interaction that does not fit into those defined above.

Response Patterns

The table below outlines the format of the strings within "correctResponsesPattern" property for each interaction type. This format is also used to represent the learner's response within the Result Object. These formats were originally based on the requirements relating to "cmi.interactions.n.correct_responses.n.pattern" as defined in the SCORM 2004 4th Edition Run-Time Environment. See Appendix C for examples of each format.

interactionType	Format
true-false	Either `true` or `false`
choice	A list of item ids delimited by `[,]`. If the response contains only one item, the delimiter MUST not be used.
fill-in and long-fill-in	A list of responses delimited by `[,]`. If the response contains only one item, the delimiter MUST not be used.
matching	A list of matching pairs, where each pair consists of a source item id followed by a target item id. Items can appear in multiple (or zero) pairs. Items within a pair are delimited by `[.]`. Pairs are delimited by `[,]`.
performance	A list of steps containing a step ids and the response to that step. Step ids are separated from responses by `[.]`. Steps are delimited by `[,]`. The response can be a String as in a fill-in interaction or a number range as in a numeric interaction.

sequencing	An ordered list of item ids delimited by `[,]`.
likert	A single item id
numeric	A range of numbers represented by a minimum and a maximum delimited by `[:]`. Where the range does not have a maximum or does not have a minimum, that number is omitted but the delimiter is still used. E.g. `[:]4` indicates a maximum for 4 and no minimum. Where the correct response or learner's response is a single number rather than a range, the single number with no delimiter MAY be used.
other	Any format is valid within this string as appropriate for the type of interaction.

Correct Responses Pattern

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

```
"correctResponsesPattern": [
    "foo[,]bar",
    "foo"
]
```

In this example, either "foo" and "bar", *or* just "foo" are correct learner responses; "bar" on it's own is not.

The Correct Responses Pattern, if used, is intended to be an exhaustive list of possible correct responses. Where the criteria for a question are complex and correct responses cannot be exhaustively listed, Learning Record Providers are discouraged from using the "correctResponsesPattern" property.

Learning Record Consumers cannot infer success based on comparison of the response with the Correct Responses Pattern, nor can they rely on the Correct Responses Pattern always being exhaustive. The Learning Record Provider is allowed to mark questions as correct where the response does not match the correct responses pattern, though this is discouraged except in exceptional circumstances.

Where the Correct Responses Pattern contains an empty array, the meaning of this is that there is no correct answer; all answers are incorrect. Where any answer is correct (e.g. in a survey), the Correct Responses Pattern property is omitted.

Characterstring parameters

Some of the values within the responses described above can be prepended with certain additional parameters. These were originally based on the characterstring delimiters defined in the SCORM 2004 4th Edition Run-Time Environment. These parameters are represented by the format `{parameter=value}`. See the long-fill-in example within Appendix C.

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

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

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

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

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

Requirements

- Interaction Activities MUST have a valid interactionType.
- Interaction Activities SHOULD have the Activity type `http://adlnet.gov/expapi/activities/cmi.interaction`.
- An LRS, upon consuming a valid interactionType, MAY validate the remaining properties as specified for Interaction Activities and MAY return `400 Bad Request` if the remaining properties are not valid for the Interaction Activity.
- The LRS SHOULD* NOT enforce character limits relating to response patterns.

- The LRS SHOULD* NOT limit the length of the correctResponsesPattern array for any interactionType.

Interaction Components

Details

Interaction components are defined as follows:

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

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

interactionType	supported interaction component list(s)	Description
choice, sequencing	choices	A list of the options available in the interaction for selection or ordering.
likert	scale	A list of the options on the likert scale.
matching	source, target	Lists of sources and targets to be matched.
performance	steps	A list of the elements making up the performance interaction.
true-false, fill-in, long-fill-in, numeric, other	[No component lists supported]	

Requirements

- Within an array of interaction components, all id values MUST be distinct.
- An interaction component's id value SHOULD NOT have whitespace.

Examples

See Appendix C for examples of Activity Definitions for each of the cmi.interaction types.

2.4.4.2 When the "Object" is an Agent or a Group

Requirements

- Statements that specify an Agent or Group as an Object MUST specify an "objectType" property.

See Actor for details regarding Agents.

2.4.4.3 When the "Object" is a Statement

Rationale

There are two possibilities for using a Statement as an Object. First, an Object can take on the form of a Statement that already exists by using a Statement Reference. A common use case for Statement References is grading or commenting on an experience that could be tracked as an independent event. The special case of voiding a Statement would also use a Statement Reference. Second, an Object can be a brand new Statement by using a SubStatement. Each type is defined below.

Statement References

Description

A Statement Reference is a pointer to another pre-existing Statement.

Requirements

- A Statement Reference MUST specify an "objectType" property with the value `StatementRef`.
- A Statement Reference MUST set the "id" property to the UUID of a Statement. There is no requirement for the LRS to validate that the UUID matches a Statement that exists.

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

Property	Type	Description	Required
objectType	String	In this case, MUST be `StatementRef`.	Required

id	UUID	The UUID of a Statement.	Required

Example

Assuming that some Statement has already been stored with the id `8f87ccde-bb56-4c2e-ab83-44982ef22df0`, the following example shows how a comment could be issued on the original Statement, using a new Statement:

```
{
        "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!"
        }
}
```

SubStatements

Description

A SubStatement is like a StatementRef in that it is included as part of a containing Statement, but unlike a StatementRef, it does not represent an event that has occurred. It can be used to describe, for example, a predication of a potential future Statement or the behavior a teacher looked for when evaluating a student (without representing the student actually doing that behavior).

Requirements

- A SubStatement MUST specify an "objectType" property with the value `SubStatement`.
- A SubStatement MUST be validated as a Statement in addition to other SubStatement requirements.
- A SubStatement MUST NOT have the "id", "stored", "version" or "authority" properties.

- A SubStatement MUST NOT contain a SubStatement of its own, i.e., cannot be nested.

Example

One interesting use of SubStatements is in creating Statements of intention. For example, using SubStatements we can create Statements of the form `"<I> <planned> (<I> <did> <this>)"` to indicate that we've planned to take some action. The concrete example that follows logically states that "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"
                                }
                        }
                }
        }
}
```

2.4.5 Result

Description

An optional property that represents a measured outcome related to the Statement in which it is included.

Details

The following table contains the properties of the Result Object.

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

2.4.5.1 Score

Description

An optional property that represents the outcome of a graded Activity achieved by an Agent.

Details

The table below defines the Score Object.

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

min	Decimal number less than max (if present)	The lowest possible score for the experience described by the Statement.	Optional
max	Decimal number greater than min (if present)	The highest possible score for the experience described by the Statement.	Optional

The properties of the Score Object are based on the corresponding properties of `cmi.score` as defined in SCORM 2004 4th Edition. The "scaled" and "raw" properties do not necessarily relate directly as scaling and normalization can be applied differently by Learning Record Providers within different Communities of Practice. Scaling and normalization are outside the scope of this specification.

Requirements

- The Score Object SHOULD include "scaled" if a logical percent based score is known.
- The Score Object SHOULD NOT be used for scores relating to progress or completion. Consider using an extension (preferably from an established Community of Practice) instead.

2.4.6 Context

Description

An optional property that provides a place to add contextual information to a Statement. All "context" properties are optional.

Rationale

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

Details

The following table contains the properties of the Context Object.

Property	Type	Description	Required

registration	UUID	The registration that the Statement is associated with.	Optional
instructor	Agent (MAY be a Group)	Instructor that the Statement relates to, if not included as the Actor of the Statement.	Optional
team	Group	Team that this Statement relates to, if not included as the Actor of the Statement.	Optional
contextActivities	contextActivities Object	A map of the types of learning activity context that this Statement is related to. Valid context types are: `parent`, `"grouping"`, `"category"` and `"other"`.	Optional
revision	String	Revision of the learning activity associated with this Statement. Format is free.	Optional
platform	String	Platform used in the experience of this learning activity.	Optional
language	String (as defined in RFC 5646)	Code representing the language in which the experience being recorded in this Statement (mainly) occurred in, if applicable and known.	Optional
statement	Statement Reference	Another Statement to be considered as context for this Statement.	Optional
extensions	Object	A map of any other domain-specific context relevant to this Statement. For example, in a flight simulator altitude, airspeed, wind, attitude, GPS coordinates might all be relevant (See Extensions)	Optional

Requirements

- The "revision" property MUST only be used if the Statement's Object is an Activity.
- The "platform" property MUST only be used if the Statement's Object is an Activity.
- The "language" property MUST NOT be used if not applicable or unknown.
- The "revision" property SHOULD be used to track fixes of minor issues (like a spelling error).

- The "revision" property SHOULD NOT be used if there is a major change in learning objectives, pedagogy, or assets of an Activity. (Use a new Activity id instead).

Note: Revision has no behavioral implications within the scope of xAPI. It is simply stored so that it is available (e.g. for interpreting and displaying data).

2.4.6.1 Registration Property

Rationale/Details

When an LRS is an integral part of an LMS, the LMS likely supports the concept of registration. The Experience API applies the concept of registration more broadly. A registration could be considered to be an attempt, a session, or could span multiple Activities. There is no expectation that completing an Activity ends a registration. Nor is a registration necessarily confined to a single Agent.

The Registration is also used when storing documents within the State Resource, e.g. for bookmarking. Normally the same registration is used for requests to both the Statement and State Resources relating to the same learning experience so that all data recorded for the experience is consistent.

2.4.6.2 ContextActivities Property

Description

A map of the types of learning activity context that this Statement is related to.

Rationale

Many Statements do not just involve one (Object) Activity that is the focus, but relate to other contextually relevant Activities. The "contextActivities" property allow for these related Activities to be represented in a structured manner.

Details

There are four valid context types. All, any or none of these MAY be used in a given Statement:

- **Parent**: an Activity with a direct relation to the Activity which is the Object of the Statement. In almost all cases there is only one sensible parent or none, not multiple. For example: a Statement about a quiz question would have the quiz as its parent Activity.

- **Grouping**: an Activity with an indirect relation to the Activity which is the Object of the Statement. For example: a course that is part of a qualification. The course has several classes. The course relates to a class as the parent, the qualification relates to the class as the grouping.
- **Category**: an Activity used to categorize the Statement. "Tags" would be a synonym. Category SHOULD be used to indicate a profile of xAPI behaviors, as well as other categorizations. For example: Anna attempts a biology exam, and the Statement is tracked using the cmi5 profile. The Statement's Activity refers to the exam, and the category is the cmi5 profile.
- **Other**: a contextActivity that doesn't fit one of the other properties. For example: Anna studies a textbook for a biology exam. The Statement's Activity refers to the textbook, and the exam is a contextActivity of type `other`.

Single Activity Objects are allowed as values so that 0.95 Statements will be compatible with 1.0.0.

Note: The values in this section are not for expressing all the relationships the Statement Object has. Instead, they are for expressing relationships appropriate for the specific Statement (though the nature of the Object will often be important in determining that). For instance, it is appropriate in a Statement about a test to include the course the test is part of as a "parent", but not to include every possible degree program the course could be part of in the grouping value.

Requirements

- Every key in the contextActivities Object MUST be one of parent, grouping, category, or other.
- Every value in the contextActivities Object MUST be either a single Activity Object or an array of Activity Objects.
- The LRS MUST return every value in the contextActivities Object as an array, even if it arrived as a single Activity Object.
- The LRS MUST return single Activity Objects as an array of length one containing the same Activity.
- The Learning Record Provider SHOULD ensure that every value in the contextActivities Object is an array of Activity Objects instead of a single Activity Object.

Example

Consider the following hierarchical structure: "Questions 1 to 6" are part of "Test 1" which in turn belongs to the course "Algebra 1". The six questions are registered as part of the test by declaring "Test 1" as their parent. Also they are grouped with other Statements about "Algebra 1" to fully mirror the hierarchy. This is particularly useful when the Object of the Statement is an Agent, not an Activity. "Andrew mentored Ben with context Algebra I".

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

2.4.7 Timestamp

Description

The time at which the experience occurred.

Details

The "timestamp" property is of type Timestamp. It is formatted according to the normal format of ISO 8601 and corresponds to the time of when the events described within this Statement occurred. If it is not included in the Statement when it is submitted to the LRS, the LRS populates it with the same value it would use with Stored.

The "timestamp property" in a Statement can differ from the "stored" property (the time at which the Statement is stored). Namely, there can be delays between the occurrence of the experience and the reception of the corresponding Statement by the LRS.

Where the experience occurs over a period of time, the "timestamp" property can represent the start, end or any point of time during the experience. It is expected that Communities of Practice will define an appropriate point to record the timestamp for different experiences. For example, when recording the experience of eating at a restaurant, it might be most appropriate to record the timestamp of the start of the experience; when recording the experience of completing a qualification, it might be most appropriate to record the timestamp of the end of the experience. These examples are for illustrative purposes only and are not meant to be prescriptive.

Requirements

- For requirements pertaining to the Timestamp data type, see Section 4.5 ISO 8601 Timestamps.
- The "timestamp" property SHOULD* be set by the LRS to the value of the "stored" property if not provided.
- A "timestamp" property MAY represent any point during an experience, not necessarily the beginning or end.
- A Learning Record Provider MUST NOT use a future time for a "timestamp" property in a Statement.
- A SubStatement MAY have a "timestamp" property that is in the future.
- An LRS SHOULD* NOT reject a timestamp for having a greater value than the current time, to prevent issues due to clock errors.

2.4.8 Stored

Description

The time at which a Statement is stored by the LRS. This can be any time between when the LRS receives the Statement and when it is written to storage.

Details

The "stored" property is of type Timestamp. The "stored" property is the literal time the Statement was stored.

Requirements

- For requirements pertaining to the Timestamp data type, see ISO 8601 Timestamps below.
- The "stored" property MUST be set by the LRS; An LRS SHOULD validate and then MUST overwrite any value currently in the "stored" property of a Statement it receives.
- The "stored" property SHOULD be the current or a past time.

2.4.9 Authority

Description

The authority property provides information about whom or what has asserted that this Statement is true.

Details

The asserting authority represents the authenticating user or some system or application.

Requirements

- Authority MUST be an Agent, except in 3-legged OAuth, where it MUST be a Group with two Agents. The two Agents represent an application and user together.
- The LRS MUST include the user as an Agent as the entire authority if a user connects directly (using HTTP Basic Authentication) or is included as part of a Group.
- The LRS MUST ensure that all Statements stored have an authority.
- The LRS SHOULD overwrite the authority on all Statements it stores, based on the credentials used to send those Statements.
- The LRS MAY leave the submitted authority unchanged but SHOULD do so only where a strong trust relationship has been established, and with extreme caution.
- The LRS MAY identify the user with any of the legal identifying properties if a user connects directly (using HTTP Basic Authentication) or a part of 3-legged OAuth.

OAuth Credentials as Authority

Description

This is a workflow for use of OAuth. 2-legged and 3-legged OAuth are both supported.

Details

This workflow assumes a Statement is stored using a validated OAuth connection and the LRS creates or modifies the authority property of the Statement.

In a 3-legged OAuth workflow, authentication involves both an OAuth consumer and a user of the OAuth service provider. For instance, requests made by an authorized Twitter plug-in on their Facebook account will include credentials that are specific not only to Twitter as a Client application, or them as a user, but the unique combination of both.

Requirements

- The authority MUST contain an Agent Object that represents the OAuth consumer, either by itself, or as part of a group in the case of 3-legged OAuth.

- The Agent representing the OAuth consumer MUST be identified by account.
- The Agent representing the OAuth consumer MUST use the consumer key as the value of the account's "name" property.
- If the Agent representing the OAuth consumer is a registered application, the token request endpoint MUST be used as the value of the account's "homePage" property.
- If the Agent representing the OAuth consumer is not a registered application, the temporary credentials endpoint MUST be used as the value of the account's "homePage" property.
- An LRS MUST NOT trust the application portion of the authority in the event the account name is from the same source as the unregistered application. (Multiple unregistered applications could choose the same consumer key. As a result, there is no consistent way to verify this combination of temporary credentials and the account name.)
- Each unregistered consumer SHOULD use a unique consumer key.

Example

The pairing of an OAuth consumer and a user.

```
"authority": {
        "objectType" : "Group",
        "member": [
                {
                        "account": {

        "homePage":"http://example.com/xAPI/OAuth/Token",
                                "name":"oauth_consumer_x75db"
                        }
                },
                {
                        "mbox":"mailto:bob@example.com"
                }
        ]
}
```

2.4.10 Version

Description

Version information in Statements helps Learning Record Consumers get their bearings. Since the Statement data model is guaranteed consistent through all 1.0.x versions, in order to support data flow among such LRSs, the LRS is given some flexibility on Statement versions that are accepted.

Requirements

- Version MUST be formatted as laid out for the API version header in Versioning

LRS Requirements

- An LRS MUST accept all Statements where their version starts with `1.0.` if they otherwise validate.
- An LRS MUST reject all Statements with a version specified that does not start with `1.0.`.
- Statements returned by an LRS MUST retain the version they are accepted with. If they lack a version, the version MUST be set to `1.0.0`.

Learning Record Provider Requirements

- If Learning Record Providers set the Statement version, they MUST set it to `1.0.0`.
- Learning Record Providers SHOULD NOT set the Statement version.

Note: The requirement for the "version" property to contain the value `1.0.0`, rather than the latest patch version is deliberate since Statements in any version of 1.0.x conform to the 1.0.0 data model. In fact, a single Statement may be included in multiple requests over time, each following a different patch version of the specification. The patch version of the specification being followed can be determined from the "X-Experience-API-Version" header being used in each request.

2.4.11 Attachments

Rationale

In some cases an Attachment is logically an important part of a Learning Record. It could be an essay, a video, etc. Another example of such an Attachment is (the image of) a certificate that was granted as a result of an experience. It is useful to have a way to store these Attachments in and retrieve them from an LRS.

Details

The table below lists all properties of the Attachment Object.

Property	Type	Description	Required	Corresponding request parameter

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

In the case of wanting to include an Attachment(s) for a SubStatement, it is strongly recommended to include the Attachment(s) in the Statement's Attachment object and to include the payloads as normally done for a Statement.

2.5 Retrieval of Statements

Description

A collection of Statements can be retrieved by performing a query on the Statement Resource, see Statement Resource for details.

Details

The following table shows the data structure for the results of queries on the Statement Resource.

Property	Type	Description	Required
statements	Array of Statements	List of Statements. If the list returned has been limited (due to pagination), and there are more results, they will be located at the "statements" property within the container located at the IRL provided by the "more" property of this Statement result Object. Where no matching Statements are found, this property will contain an empty array.	Required
more	IRL	Relative IRL that can be used to fetch more results, including the full path and optionally a query string but excluding scheme, host, and port. Empty string if there are no more results to fetch.	Required if the list returned has been limited, otherwise optional.

Requirements

- The IRL retrieved from the "more" property MUST be usable for at least 24 hours after it is returned by the LRS.
- An LRS MAY include all necessary information within the "more" property IRL to continue the query to avoid the need to store IRLs and associated query data.
- An LRS SHOULD NOT generate extremely long IRLs within the "more" property.
- An LRS MAY re-run the query at the point in time that the IRL retrieved from the "more" property is accessed such that the batch retrieved includes Statements which would have been included in that batch if present in the LRS at the time the original query was run and excludes Statements from that batch which have since been voided.
- Alternatively, an LRS MAY cache a list of Statements to be returned at the "more" property such that the batch of Statements returned matches those Statements that would have been returned when the original query was run.
- An LRS MAY remove voided Statements from the cached list of Statements if using this method.
- A Learning Record Consumer SHOULD NOT attempt to interpret any meaning from the IRL returned from the "more" property.

2.6 Signed Statements

Description

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

Rationale

Some Statements will have regulatory or legal significance, or otherwise require strong and durable evidence of their authenticity and integrity. It might be necessary to verify these Statements without trusting the environment they were first recorded in, or perhaps without access to that environment. Digital signatures will enable a third-party to validate such Statements.

Details

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

Signature Requirements

- A Signed Statement MUST include a JSON web signature (JWS) as defined here: http://tools.ietf.org/html/rfc7515, as an Attachment with a usageType of `http://adlnet.gov/expapi/attachments/signature` and a contentType of `application/octet-stream`.
- JWS Compact Serialization SHOULD* be used to create the JSON web signature. Use of JWS JSON Serialization is strongly discouraged, is unlikely to be interoperble with other systems, and will be forbidden in a future version of this specification.
- The JWS signature MUST have a payload of a valid JSON serialization of the complete Statement before the signature was added.
- The JWS signature MUST use an algorithm of "RS256", "RS384", or "RS512".
- The JWS signature SHOULD have been created based on the private key associated with an X.509 certificate.
- If X.509 was used to sign, the JWS header SHOULD include the "x5c" property containing the associated certificate chain.

LRS Requirements

- The LRS MUST reject requests to store Statements that contain malformed signatures, with `400 Bad Request`.
- The LRS SHOULD include a message in the response of a rejected statement.
- In order to verify signatures are well formed, the LRS MUST do the following:
    - Decode the JWS signature, and load the signed serialization of the Statement from the JWS signature payload.
    - Validate that the original Statement is logically equivalent to the received Statement. See Statement comparision requirements.
    - If the JWS header includes an X.509 certificate, validate the signature against that certificate as defined in JWS.
    - Validate that the signature requirements outlined above have been met.

Note: The step of validating against the included X.509 certificate is intended as a way to catch mistakes in the signature, not as a security measure. The steps to authenticate a signed Statement will vary based on the degree of certainty required and are outside the scope of this specification.

Client Requirements

- Clients MUST follow the signature requirements outlined above.
- Clients MUST NOT assume a signature is valid simply because an LRS has accepted it.

Example

See Appendix D: Example Signed Statement for an example.

3.0 Metadata

Metadata is additional information about the resource. It enables decision making, search, and discoverability. In xAPI, metadata can be utilized in a variety of locations. The most common is within Activity Definitions.

3.1 IRI Requirements

xAPI uses IRIs for identifiers. Using IRIs ensures uniqueness and promotes resolvability. The LRS and Learning Record Provider each have responsibilities in regard to each IRI as outlined below. Activity Definitions have additional rules which can be found here.

Metadata Provider Requirements

These requirements also apply to Learning Record Providers defining new IRIs.

- Metadata Providers defining new IRIs SHOULD* only use IRIs they control or have permission from the controller to use.
- Metadata Providers defining new Verb IRIs MUST only use IRIs they control or have permission from the controller to use.
- Where a suitable identifier already exists, the Metadata Provider SHOULD use the corresponding existing identifier and SHOULD NOT create a new identifier.
- When re-using an existing identifier, Metadata Providers SHOULD* ensure that the exact character equivelent IRI is used.
- The Metadata Provider MAY create their own identifiers where a suitable identifier does not already exist.
- When defining identifiers, the Metadata Provider MAY use IRIs containing anchors so that a single page can contain definitions for multiple identifiers. E.g. `http://example.com/xapi/verbs#defenestrated`
- When defining identifiers, the Metadata Provider SHOULD use lowercase IRIs.

LRS Requirements

- When storing or comparing IRIs, LRSs SHOULD* handle them only by using one or more of the approaches described in 5.3.1 (Simple String Comparison) and 5.3.2 (Syntax-Based Normalization) of RFC 3987, and SHOULD* NOT handle them using any approaches described in 5.3.3 (Scheme-Based Normalization) or 5.3.4 (Protocol-Based Normalization) of the same RFC, or any other approaches.
- LRSs SHOULD* apply the same IRI comparison and normalization rules with all IRIs in parameters and fields defined to contain IRIs.

3.2 Hosted Metadata

Description

Additional information about an identifier can be provided within a Statement and can be hosted at the location pointed to by the identifier IRI. Including metadata in a Statement allows metadata about the IRI to be expressed without the necessity of resolving it. Hosting metadata at the IRI location allows the owner of the IRI to define the canonical metadata for that IRI.

Details

There are several types of IRI identifiers used in this specification:

- Verb
- Activity id
- Activity type
- extension key
- attachment usage type

For the structure of hosted metadata about Activity ids, see Activity Definition Object.

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

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

Hosted metadata consists of a document containing a JSON object as described above. If this hosted metadata is provided, it is the canonical source of information about the identifier it describes. It is recommended that those implementing xAPI look for and use established, widely adopted identifiers for all types of IRI identifiers other than Activity id.

Metadata Provider Requirements

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

LRS Requirements

- The LRS MAY act as a Metadata Consumer and attempt to resolve identifier IRIs.

- If an Activity IRI is a URL, an LRS SHOULD attempt to GET that URL, and include in HTTP headers: `Accept: application/json, */*`. This SHOULD be done as soon as practical after the LRS first encounters the Activity id.
- Upon loading JSON which is a valid Activity Definition from a URL used as an Activity id, an LRS SHOULD incorporate the loaded definition into its canonical definition for that Activity, while preserving names or definitions not included in the loaded definition.
- Upon loading any document from which the LRS can parse an Activity Definition from a URL used as an Activity id, an LRS MAY consider this definition when determining its canonical representation of that Activity's definition.

Metadata Consumer Requirements

- If a Metadata Consumer obtains metadata from an IRI, it SHOULD make a strong presumption that the metadata found at that IRI is authoritative in regards to the properties and languages included in that metadata.
- The Metadata Consumer MAY use other sources of information to fill in missing details, such as translations, or take the place of the hosted metadata entirely if it was not provided, cannot be loaded or the Metadata Consumer does not trust it. Other sources of information MAY include metadata in other formats stored at the IRI of an identifier, particularly if that identifier was not coined for use with this specification.

4.0 Special Data Types and Rules

The following are data types requiring additional rules that are found commonly in this specification.

4.1 Extensions

Description

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

Details

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

Requirements

- The keys of an extensions map MUST be IRIs.
- An LRS MUST NOT reject a Statement based on the values of the extensions map.
- Learning Record Providers SHOULD always strive to map as much information as possible into the built-in elements in order to leverage interoperability among Experience API conformant tools.
- All extension IRIs SHOULD have controllers.
- The controller of an IRL extension key SHOULD make a human-readable description of the intended meaning of the extension supported by the IRL accessible at the IRL.

Note: A Statement defined entirely by its extensions becomes meaningless as no other system can make sense of it.

4.2 Language Maps

Description

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

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

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

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

4.3 IRIs

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

See Metadata.

4.4 UUIDs

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

4.5 ISO 8601 Timestamps

Timestamps are a format type which represent a specific time. They are formatted according to ISO 8601's normal format. Statements sent to an LRS can be expected to keep precision to at least milliseconds

Requirements

- A Timestamp MUST be formatted according to ISO 8601.
- A Timestamp SHOULD* be expressed using the format described in RFC 3339, which is a profile of ISO 8601.
- A Timestamp MUST preserve precision to at least milliseconds (3 decimal points beyond seconds).
- A Timestamp SHOULD* include the time zone.

- If the Timestamp includes a time zone, the LRS MAY be return the Timestamp using a different timezone to the one originally used in the Statement so long as the point in time referenced is not affected.
- The LRS SHOULD* return the Timestamp in UTC timezone.
- A Timestamp MAY be truncated or rounded to a precision of at least 3 decimal digits for seconds.

4.6 ISO 8601 Durations

Durations are strings representing the amount of time something took.

Requirements

- A Duration MUST be expressed using the format for Duration in ISO 8601:2004(E) section 4.4.3.2. The alternative format (in conformity with the format used for time points and described in ISO 8601:2004(E) section 4.4.3.3) MUST NOT be used.
- Learning Record Providers SHOULD provide a maximum precision of 0.01 seconds.
- Learning Record Providers MAY provide less precision, for example in the case of reading a University Degree precision might be in months or years.
- On receiving a Duration with more than 0.01 second precision, the LRS SHOULD* NOT reject the request but MAY truncate the "duration" property to 0.01 second precision.
- When comparing Durations, any precision beyond 0.01 second precision SHOULD* NOT be included in the comparison.

Examples

The table below provides some example ISO 8601 Durations. This list is not intended to be exhaustive.

Example	Explanation
PT4H35M59.14S	Four hours, thirty five minutes and 59.14 seconds.
PT16559.14S	The same time period as above represented in seconds. (Note: if the time period in question contained a leap second, this conversion would be inaccurate)
P3Y1M29DT4H35M59.14S	A Duration also including years, months and days.

P3Y	Approximately three years e.g. completion of a qualification.
P4W	Four weeks. Note that weeks cannot be combined with other time periods. 'P4W1D' is not valid.

Durations are expected to be presented in the format in which they are recorded. For example if a Duration is tracked in seconds (or fractions of a second) there is no need to convert this to hours, minutes, and seconds.

Appendices

Appendix A: Example Statements

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

```
{
        "id":"fd41c918-b88b-4b20-a0a5-a4c32391aaa0",
        "timestamp": "2015-11-18T12:17:00+00:00",
        "actor":{
                "objectType": "Agent",
                "name":"Project Tin Can API",
                "mbox":"mailto:user@example.com"
        },
        "verb":{
                "id":"http://example.com/xapi/verbs#sent-a-statement",
                "display":{
                        "en-US":"sent"
                }
        },
        "object":{
                "id":"http://example.com/xapi/activity/simplestatement",
                "definition":{
                        "name":{
                                "en-US":"simple statement"
                        },
                        "description":{
                                "en-US":"A simple Experience API statement.
Note that the LRS
                                does not need to have any prior information
about the Actor (learner), the
                                verb, or the Activity/object."
                        }
                }
        }
}
```

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

```
{
        "id":"7ccd3322-e1a5-411a-a67d-6a735c76f119",
        "timestamp": "2015-12-18T12:17:00+00:00",
        "actor":{
        "objectType": "Agent",
                "name":"Example Learner",
                "mbox":"mailto:example.learner@adlnet.gov"
        },
        "verb":{
                "id":"http://adlnet.gov/expapi/verbs/attempted",
                "display":{
                        "en-US":"attempted"
                }
        },
        "object":{
                "id":"http://example.adlnet.gov/xapi/example/simpleCBT",
                "definition":{
                        "name":{
                                "en-US":"simple CBT course"
                        },
                        "description":{
                                "en-US":"A fictitious example CBT course."
                        }
                }
        },
        "result":{
                "score":{
                        "scaled":0.95
                },
                "success":true,
                "completion":true,
                "duration": "PT1234S"
        }
}
```

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

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

"http://example.com/profiles/meetings/resultextensions/minuteslocation":
"X:\\meetings\\minutes\\examplemeeting.one"
        },
        "success": true,
        "completion": true,
        "response": "We agreed on some example actions.",
        "duration": "PT1H0M0S"
    },
    "context": {
        "registration": "ec531277-b57b-4c15-8d91-d292c5b2b8f7",
        "contextActivities": {
            "parent": [
                {
                    "id": "http://www.example.com/meetings/series/267",
                    "objectType": "Activity"
                }
            ],
            "category": [
                {
                    "id":
"http://www.example.com/meetings/categories/teammeeting",
                    "objectType": "Activity",
                    "definition": {
                                   "name": {
                                       "en": "team meeting"
                                   },
                                   "description": {
                                       "en": "A category of meeting used for
regular team meetings."
                                   },
                                   "type":
"http://example.com/expapi/activities/meetingcategory"
                               }
                }
            ],
            "other": [
                {
                    "id": "http://www.example.com/meetings/occurances/34257",
                    "objectType": "Activity"
                },
                {
                    "id":
"http://www.example.com/meetings/occurances/3425567",
                    "objectType": "Activity"
                }
            ]
        },
        "instructor" :
        {
               "name": "Andrew Downes",
            "account": {
                "homePage": "http://www.example.com",
                "name": "13936749"
            },
            "objectType": "Agent"
        },
        "team":
        {
               "name": "Team PB",
               "mbox": "mailto:teampb@example.com",
               "objectType": "Group"
        },
        "platform" : "Example virtual meeting software",
        "language" : "tlh",
        "statement" : {
               "objectType":"StatementRef",
               "id" :"6690e6c9-3ef0-4ed3-8b37-7f3964730bee"
        }

    },
    "timestamp": "2013-05-18T05:32:34.804+00:00",
    "stored": "2013-05-18T05:32:34.804+00:00",
    "authority": {
        "account": {
            "homePage": "http://cloud.scorm.com/",
            "name": "anonymous"
        },
        "objectType": "Agent"
    },
    "version": "1.0.0",
    "object": {
        "id": "http://www.example.com/meetings/occurances/34534",
        "definition": {
            "extensions": {

"http://example.com/profiles/meetings/activitydefinitionextensions/room":
{"name": "Kilby", "id" : "http://example.com/rooms/342"}
            },
            "name": {
                "en-GB": "example meeting",
                "en-US": "example meeting"
            },
            "description": {
                "en-GB": "An example meeting that happened on a specific
occasion with certain people present.",
                "en-US": "An example meeting that happened on a specific
occasion with certain people present."
            },
            "type": "http://adlnet.gov/expapi/activities/meeting",
            "moreInfo": "http://virtualmeeting.example.com/345256"
        },
        "objectType": "Activity"
    }
}
```

Appendix B: Examples of Statement's Objects of different types

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

Object is Activity

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

Object is Agent

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

Object is Group

This example shows an Identified Group with members.

```
{
    "name": "Example Group",
    "account" : {
        "homePage" : "http://example.com/homePage",
        "name" : "GroupAccount"
    },
    "objectType": "Group",
    "member": [
        {
            "name": "Andrew Downes",
            "mbox": "mailto:andrew@example.com",
            "objectType": "Agent"
        },
        {
            "name": "Aaron Silvers",
            "openid": "http://aaron.openid.example.org",
            "objectType": "Agent"
        }
    ]
}
```

Object is Statement

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

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

Appendix C: Example definitions for Activities of type `cmi.interaction`

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 experiences."
        ]
}
```

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[:]"
        ]
}
```

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

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)"
        ]
}
```

Appendix D: Example Signed Statement

An example signed Statement, as described in: Section 2.6 Signed Statements.

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

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

Example private key for X.509 certificate that will be used for signing:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDjxvZXF30WL4oKjZYXgR0ZyaX+u3y6+JqTqiNkFa/VTnet6Ly2
OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkW13j06j5iNn4obcCVWTL9yXNvJH+Ko+xu4Y
l/ySPRrIPyTjtHdG0M2XzIlmmLqm+CAS+KCbJeH4tf543kIWC5pC5p3cVQIDAQAB
AoGAOejdvGq2XKuddu1kWXl0Aphn4YmdPpPyCNTaxplU6PBYMRjY0aNgLQE6bO2p
/HJiU4Y4PkgzkEgCu0xf/mOq5DnSkX32ICoQS6jChABAe20ErPfm5t8h9YKsTfn9
40lAouuwD9ePRteizd4YvHtiMMwmh5PtUoCbqLefawNApAECQQD1mdBW3zL0okUx
2pc4tttn2qArCG4CsEZMLlGRDd3FwPWJz3ZPNEEgZWXGSpA9F1QTZ6JYXIfejjRo
UuvRMWeBAkEA7WvzDBNcv4N+xeUKvH8ILti/BM58LraTtqJlzjQSovek0srxtmDg
5of+xrxN6IM4p7yvQa+7YOUOukrVXjG+1QJBAI2mBrjzxgm9xTa5odn97JD7UMFA
/WHjlMe/Nx/35V52qaav1sZbluw+TvKMcqApYj5G2SUpSNudHLDGkmd2nQECQFfc
lBRK8g7ZncekbGW3aRLVGVOxClnLLTzwOlamBKOUm8V6XxsMHQ6TE2D+fKJoNUY1
2HGpk+FWwy2D1hRGuoUCQAXfaLSxtaWdPtlZTPVueF7ZikQDsVg+vtTFgpuHloR2
6EVc1RbHHZm32yvGDY8IkcoMfJQqLONDdLfS/05yoNU=
-----END RSA PRIVATE KEY-----
```

Example public X.509 certificate

```
-----BEGIN CERTIFICATE-----
MIIDATCCAmqgAwIBAgIJAMB1csNuA6+kMA0GCSqGSIb3DQEBBQUAMHExCzAJBgNV
BAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUgQ29t
```

```
cGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBsZUBl
eGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI4MzBaFw0xNDA0MDQxNTI4MzBaMIGWMQsw
CQYDVQQGEwJVUzESMBAGA1UECBMJVGVubmVzc2VlMREwDwYDVQQHEwhGcmFua2xp
bjEYMBYGA1UEChMPRXhhbXBsZSBDb21wYW55MRAwDgYDVQQLEwdFeGFtcGxlMRAw
DgYDVQQDEwdFeGFtcGxlMSIwIAYJKoZIhvcNAQkBFhNleGFtcGxlQGV4YW1wbGUu
Y29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDjxvZXF30WL4oKjZYXgR0Z
yaX+u3y6+JqTqiNkFa/VTnet6Ly2OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkW13j06j
5iNn4obcCVWTL9yXNvJH+Ko+xu4Yl/ySPRrIPyTjtHdG0M2XzIlmmLqm+CAS+KCb
JeH4tf543kIWC5pC5p3cVQIDAQABo3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIB
DQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUVs3v
5afEdOeoYeVajAQE4v0WS1QwHwYDVR0jBBgwFoAUyVIc3yvra4EBz20I4BF39IAi
xBkwDQYJKoZIhvcNAQEFBQADgYEAgS/FF5D0Hnj44rvT6kgn3kJAvv2lj/fyjztK
IrYS33ljXGn6gGyA4qtbXA23PrO4uc/wYCSDICDpPobh62xTCd9qObKhgwWOi05P
SBLqUu3mwfAe15LJBJBqPVZ4K0kppePBU8m6aIZoH57L/9t4OoaL8yKs/qjKFeI1
OFWZxvA=
-----END CERTIFICATE-----
```

Example certificate authority certificate

```
-----BEGIN CERTIFICATE-----
MIIDNzCCAqCgAwIBAgIJAMB1csNuA6+jMA0GCSqGSIb3DQEBBQUAMHExCzAJBgNV
BAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUgQ29t
cGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBsZUBl
eGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI1NTNaFw0yMzA0MDIxNTI1NTNaMHExCzAJ
BgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUg
Q29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBs
ZUBleGFtcGxlLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA1sBnBWPZ
0f7WJUFTJy5+01SlS5Z6DDD6Uye9vK9AycgV5B3+WC8HC5u5h91MujAC1ARPVUOt
svPRs45qKNFIgIGRXKPAwZjawEI2sCJRSKV47i6B8bDv4WkuGvQaveZGI0qlmN5R
1Eim2gUItRj1hgcC9rQavjlnFKDY2rlXGukCAwEAAaOB1jCB0zAdBgNVHQ4EFgQU
yVIc3yvra4EBz20I4BF39IAixBkwgaMGA1UdIwSBmzCBmIAUyVIc3yvra4EBz20I
4BF39IAixBmhdaRzMHExCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUx
GDAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAG
CSqGSIb3DQEJARYTZXhhbXBsZUBleGFtcGxlLmNvbYIJAMB1csNuA6+jMAwGA1Ud
EwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEADhwTebGk735yKhm8DqCxvNnEZ0Nx
sYEYOjgRG1yXTlW5pE691fSH5AZ+T6fpwpZcWY5QYkoN6DnwjOxGkSfQC3/yGmcU
DKBPwiZ5O2s9C+fE1kUEnrX2Xea4agVngMzR8DQ6oOauLWqehDB+g2ENWRLoVgS+
ma5/Ycs0GTyrECY=
-----END CERTIFICATE-----
```

JWS Header. Note that along with specifying the algorithm, the certificate chain for the signing certificate has been included.

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

"MIIDATCCAmqgAwIBAgIJAMB1csNuA6+kMA0GCSqGSIb3DQEBBQUAMHExCzAJBgNVBAYTAlVTMRIw
EAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhb
XBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBsZUBleGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI4MzBaFw
0xNDA0MDQxNTI4MzBaMIGWMQswCQYDVQQGEwJVUzESMBAGA1UECBMJVGVubmVzc2VlMREwDwYDVQQ
HEwhGcmFua2xpbjEYMBYGA1UEChMPRXhhbXBsZSBDb21wYW55MRAwDgYDVQQLEwdFeGFtcGxlMRAw
DgYDVQQDEwdFeGFtcGxlMSIwIAYJKoZIhvcNAQkBFhNleGFtcGxlQGV4YW1wbGUuY29tMIGfMA0GC
SqGSIb3DQEBAQUAA4GNADCBiQKBgQDjxvZXF30WL4oKjZYXgR0ZyaX+u3y6+JqTqiNkFa/VTnet6L
y2OT6ZmmcJEPnq3UnewpHoOQ+GfhhTkW13j06j5iNn4obcCVWTL9yXNvJH+Ko+xu4Yl/ySPRrIPyT
jtHdG0M2XzIlmmLqm+CAS+KCbJeH4tf543kIWC5pC5p3cVQIDAQABo3sweTAJBgNVHRMEAjAAMCwG
CWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUVs3v5
afEdOeoYeVajAQE4v0WS1QwHwYDVR0jBBgwFoAUyVIc3yvra4EBz20I4BF39IAixBkwDQYJKoZIhv
cNAQEFBQADgYEAgS/FF5D0Hnj44rvT6kgn3kJAvv2lj/fyjztKIrYS33ljXGn6gGyA4qtbXA23PrO
4uc/wYCSDICDpPobh62xTCd9qObKhgwWOi05PSBLqUu3mwfAe15LJBJBqPVZ4K0kppePBU8m6aIZo
H57L/9t4OoaL8yKs/qjKFeI1OFWZxvA=",

"MIIDNzCCAqCgAwIBAgIJAMB1csNuA6+jMA0GCSqGSIb3DQEBBQUAMHExCzAJBgNVBAYTAlVTMRIw
EAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhb
XBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBsZUBleGFtcGxlLmNvbTAeFw0xMzA0MDQxNTI1NTNaFw
0yMzA0MDIxNTI1NTNaMHExCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxGDAWBgNVBAo
TD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJARYTZXhhbXBs
ZUBleGFtcGxlLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA1sBnBWPZ0f7WJUFTJy5+0
1SlS5Z6DDD6Uye9vK9AycgV5B3+WC8HC5u5h91MujAC1ARPVUOtsvPRs45qKNFIgIGRXKPAwZjawE
I2sCJRSKV47i6B8bDv4WkuGvQaveZGI0qlmN5R1Eim2gUItRj1hgcC9rQavjlnFKDY2rlXGukCAwE
AAaOB1jCB0zAdBgNVHQ4EFgQUyVIc3yvra4EBz20I4BF39IAixBkwgaMGA1UdIwSBmzCBmIAUyVIc
3yvra4EBz20I4BF39IAixBmhdaRzMHExCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlUZW5uZXNzZWUxG
DAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEQMA4GA1UEAxMHRXhhbXBsZTEiMCAGCSqGSIb3DQEJAR
YTZXhhbXBsZUBleGFtcGxlLmNvbYIJAMB1csNuA6+jMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQE
FBQADgYEADhwTebGk735yKhm8DqCxvNnEZ0NxsYEYOjgRG1yXTlW5pE691fSH5AZ+T6fpwpZcWY5Q
YkoN6DnwjOxGkSfQC3/yGmcUDKBPwiZ5O2s9C+fE1kUEnrX2Xea4agVngMzR8DQ6oOauLWqehDB+g
2ENWRLoVgS+ma5/Ycs0GTyrECY="
     ]
}
```

JWS signature

```
ew0KICAgICJhbGciOiAiUlMyNTYiLA0KICAgICJ4NWMiOiBbDQogICAgICAgICJNSUlEQVRDQ0Ftc
WdBd0lCQWdJSkFNQjFjc051QTYra01BMEdDU3FHU0liM0RRRUJCUVVBTUhFeEN6QUpCZ05WQkFZVE
FsVlRNUkl3RUFZRFZRUUlFd2xVWlc1dVpYTnpaV1V4R0RBV0JnTlZCQW9URDBWNFlXMXdiR1VnUTI
5dGNHRnVlVEVRTUE0R0ExVUVBeE1IUlhoaGJYQnNaVEVpTUNBR0NTcUdTSWIzRFFFSkFSWVRaWGho
YlhCc1pVQmxlR0Z0Y0d4bExtTnZiVEFlRncweE16QTBNRFF4TlRJNE16QmFGdzB4TkRBME1EUXhOV
Ek0TXpCYU1JR1dNUXN3Q1FZRFZRUUdFd0pWVXpFU01CQUdBMVVFQ0JNSlZHVnVibVZ6YzJWbE1SRX
dEd1lEVlFRSEV3aEdjbUZ1YTJ4cGJqRVlNQllHQTFVRUNoTVBSWGhoYlhCc1pTQkRiMjF3WVc1NU1
SQXdEZ1lEVlFRTEV3ZEZlR0Z0Y0d4bE1SQXdEZ1lEVlFRREV3ZEZlR0Z0Y0d4bE1TSXdJQVlKS29a
SWh2Y05BUWtCRmhObGVHRnRjR3hsUUdWNFlXMXdiR1V1WTI5dE1JR2ZNQTBHQ1NxR1NJYjNEUUVCQ
VFVQUE0R05BRENCaVFLQmdRRGp4dlpYRjMwV0w0b0tqWllYZ1IwWnlhWCt1M3k2K0pxVHFpTmtGYS
9WVG5ldDZMeTJPVDZabW1jSkVQbnEzVW5ld3BIb09RK0dmaGhUa1cxM2owNmo1aU5uNG9iY0NWV1R
MOXlYTnZKSCtLbyt4dTRZbC95U1BSScklQeVRqdEhkRzBNMlh6SWxtbUxxbStDQVMrS0NiSmVINHRm
NTQza0lXQzVwQzVwM2NWUUlEQVFBQm8zc3dlVEFKQmdOVkhSTUVBakFBTUN3R0NXQ0dTQUdHK0VJQ
kRRUWZGaDFQY0dWdVUxTk1JRWRsYm1WeVlYUmxaQ0JEWlhKMGFXWnBZMkYwWlRBZEJnTlZIUTRFRm
dRVVZzM3Y1YWZFE9lb1llVmFqQVFFNHYwV1MxUXdId1lEVlIwakJCZ3dGb0FVeVZJYzN5dnJhNEV
CejIwSTRCRjM5SUFpeEJrd0RRWUpLb1pJaHZjTkFRRUZCUUFEZ1lFQUdGQWdTLzdSQ0sydk1BQ2
a2duM2tKQXZ2MmxqL2Z5anp0S0lyWVMzM2xqWEduNmdHeUE0cXRiWEEyM1ByTzR1Yy93WUNTRElDR
HBQb2JoNjJ4VENkOXFPYktoZ3dXT2kwNVBTQkxxVXUzbXdmQWUxNUxKQkpCcVBWWjRLMGtwcGVQQl
U4bTZhSVpvSDU3TC85dDRPb2FMOHlLcy9xaktGZUkxT0ZXWnh2QT0iLA0KICAgICAgICAiTUlJRE5
6Q0NBcUNnQXdJQkFnSUpBTUIxY3NOdUE2K2pNQTBHQ1NxR1NJYjNEUUVCQlFVQU1IRXhDekFKQmdO
VkJBWVRBbFZUTVJJd0VBWURWUVFJRXdsVVpXNXVaWE56WldVeEdEQVdCZ05WQkFvVEQwVjRZVzF3Y
kdVZ1EyOXRjR0Z1ZVRFUU1BNEdBMVVFQXhNSFJYaGhiWEJzWlRFaU1DQUdDU3FHU0liM0RRRUpBUl
lUWlhoaGJYQnNaVUJsZUdGdGNHeGxMbU52YlRBZUZ3MHhNekEwTURReE5USTFOVE5hRncweU16QTB
NREl4TlRJMU5UTmFNSEV4Q3pBSkJnTlZCQVlUQWxWVE1SSXdFQVlEVlFRSUV3bFVaVzV1WlhOelpX
VXhHREFXQmdOVkJBb1REMFY0WVcxd2JHVWdRMjl0Y0dGdWVURVFNQTRHQTFVRUF4TUhSWGhoYlhCc
1pURWlNQ0FHQ1NxR1NJYjNEUUVKQVJZVFpYaGhiWEJzWlVCbGVHRnRjR3hsTG1OdmJUQ0JuekFOQm
drcWhraUc5dzBCQVFFRkFBT0JqUUF3Z1lrQ2dZRUExc0JuQldQWjBmN1dKVUZUSnk1KzAxU2xTNVo
2REVERENlV5ZTl2SzlBeWNnVjVCMytXQzhIQzV1NWg5MU11akFDMUFSUFZVT3RzdlBSczQ1cUtORklnSUdSWEtQQXdaamF3RUkyc0NKUlNLVjQ3aTZCOGJEdjRXa3VHdlFhdmVaR0kwcWxtTjVSMUVpbTJnVUl0UmoxaGdjQzlyUWF2amxuRktEWTJybFhHdWtDQXdFQUFhT0IxakNCMHpBZEJnTlZIUTRFRmdRVXlWSWMzeXZyYTRFQnoyMEk0QkYzOUlBaXhCa3dnYU1HQTFVZEl3U0JtekNCbUlBVXlWSWMzeXZyYTRFQnoyMEk0QkYzOUlBaXhCbWhkYVJ6TUhFeEN6QUpCZ05WQkFZVEFsVlRNUkl3RUFZRFZRUUlFd2xVWlc1dVpYTnpaV1V4RURBT0JnTlZCQWNUQjBaeVlXNXJiR2x1TVJnd0ZnWURWUVFLRXc5RmVHRnRjR3hsSUVOdmJYQmhibmt4RURBT0JnTlZCQXNUQjBWNFlXMXdiR1V4RURBT0JnTlZCQU1UQjBWNFlXMXdiR1V4SWpBZ0Jna3Foa2lHOXcwQkNRRVdFMlY0WVcxd2JHVkFaWGhoYlhCc1pTNWpiMjB3Z1o4d0RRWUpLb1pJaHZjTkFRRUJCUUFEZ1kwQU1JR0pBb0dCQU9QRzlsY1hmUll2aWdxTmxoZUJIUm5KcGY2N2ZMcjRtcE9xSTJRVnI5Vk9kNjNvdkxZNVBwbWFad2tRK2VyZFNkN0NrZWc1RDRaK0dGT1JiWGZQVHFQbUkyZmlodHdKVlpNdjNKYzI4a2Y0cWo3RzdoaVg5SkE5R1hKTk8wZDBiUXpaZk1pV2FZdXFiNElCTDRvSnNsNGZpMS9uamVRaFlMbWtMbW5keFZBZ01CQUFHamV6QjVNQWtHQTFVZEV3UUNNQUF3TEFZSllJWklBWWI0UWdFTkJCOFdIVTl3Wlc1VFUwd2dSMlZ1WlhKaGRHVmtJRU5sY25ScFptbGpZWFJsTUIwR0ExVWREZ1FXQkJSYldhWUtVWXl4SlhzbTA1REpDbDVmNzlQbVVUQWZCZ05WSFNNRUdEQVdnQlRKVWh6ZktldHJnUUhQYlFqZ0VYZjBnQ0xFR1RBTkJna3Foa2lHOXcwQkFRVUZBQU9CZ1FBWUNiUnlMUnJwb2FRTERUaHhjSFJvWnBwUFpTc001dmtYK3NYSnlRVWJVZ3RNRFcvUnh1OEZnc0Nmbm5WL1o2QjlaSS9hQ3lCWm1ZR3dtcDdDMzNPNHB4NlhIYkR0RklSL3Y5Wkh3cGZQQ3dvNVZZZGRrcDNwL0JIU05hOGFNQzZnMXFKQ2M1b1RYcXdSZGlmcDNkUzVuVTNPeUxZOUlVZGtoUEt5U1JKL0hROVN1YzVBRk1uQTh1djlOYkM3bi83R0QwV2dwVDRHWVI3UXJnWGNrTVNRcmJhVTg5U2lWM0xEdjhvMk1QWnZDa1BQSFQ2bWVnTFQrbGs2Y0pPbDFBRzdzMkduYkE4bXBtbmpsNEVhWC9YZUgrQXRuSWRpSjRUVDhmaHhmb2FxNWVFR3Jtd1prSUNYd2JKMHhXVFJuUWZQeXZOVVp6SXUrK2ZFUmJNbjJMcnc0NUV4WlNXTWMQ
```

```
Wlc1dVpYTnpaV1V4R0RBV0JnTlZCQW9URDBWNFlXMXdiR1VnUTI5dGNHRnVlVEVRTUE0R0ExVUVBe
E1IUlhoaGJYQnNaVEVpTUNBR0NTcUdTSWIzRFFFSkFSWVRaWGhoYlhCc1pVQmxlR0Z0Y0d4bExtTn
ZiWUlKQU1CMWNzTnVBNitqTUF3R0ExVWRFd1FGTUFNQkFmOHdEUVlKS29aSWh2Y05BUUVGQlFBRGd
ZRUFEaHdUZWJHazczNXlLaG04RHFDeHZObkVaME54c1lFWU9qZ1JHMXlYVGxXNXBFNjkxZlNINUFa
K1Q2ZnB3cFpjV1k1UVlrb042RG53ak94R2tTZlFDMy95R21jVURLQlB3aVo1TzJzOUMrZkUxa1VFb
nJYMlhlYTRhZ1ZuZ016UjhEUTZvT2F1TFdxZWhEQitnMkVOV1JMb1ZnUyttYTUvWWNzMEdUeXJFQ1
k9Ig0KICAgIF0NCn0.ew0KICAgICJ2ZXJzaW9uIjogIjEuMC4wIiwNCiAgICAiaWQiOiAiMzNjZmY
0MTYtZTMzMS00YzlkLTk2OWUtNTM3M2ExNzU2MTIwIiwNCiAgICAiYWN0b3IiOiB7DQogICAgICAg
ICJtYm94IjogIm1haWx0bzpleGFtcGxlQGV4YW1wbGUuY29tIiwNCiAgICAgICAgIm5hbWUiOiAiR
XhhbXBsZSBMZWFybmVyIiwNCiAgICAgICAgIm9iamVjdFR5cGUiOiAiQWdlbnQiDQogICAgfSwNCi
AgICAidmVyYiI6IHsNCiAgICAgICAgImlkIjogImh0dHA6Ly9hZGxuZXQuZ292L2V4cGFwaS92ZXJ
icy9leHBlcmllbmNlZCIsDQogICAgICAgICJkaXNwbGF5Ijogew0KICAgICAgICAgICAgImVuLVVT
IjogImV4cGVyaWVuY2VkIg0KICAgICAgICB9DQogICAgfSwNCiAgICAib2JqZWN0Ijogew0KICAgI
CAgICAiaWQiOiAiaHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g_dj14aDRrSWlIM1NtOCIsDQ
ogICAgICAgICJvYmplY3RUeXBlIjogIkFjdGl2aXR5IiwNCiAgICAgICAgImRlZmluaXRpb24iOiB
7DQogICAgICAgICAgICAibmFtZSI6IHsNCiAgICAgICAgICAgICAgICAiZW4tVVMiOiAiVGF4IFRp
cHMgJiBJbmZvcm1hdGlvbiA6IEhvdyB0byBGaWxlIGEgVGF4IFJldHVybiAiDQogICAgICAgICAgI
CB9LA0KICAgICAgICAgICAgImRlc2NyaXB0aW9uIjogew0KICAgICAgICAgICAgICAgICJlbi1VUy
I6ICJGaWxpbmcgYSB0YXggcmV0dXJuIHdpbGwgcmVxdWlyZSBmaWxsaW5nIG91dCBlaXRoZXIgYSA
xMDQwLCAxMDQwQSBvciAxMDQwRVogZm9ybSINCiAgICAgICAgICAgIH0NCiAgICAgICAgfQ0KICAg
IH0sDQogICAgInRpbWVzdGFtcCI6ICIyMDEzLTA0LTAxVDEyOjAwOjAwWiINCn0.FWuwaPhwUbkk7
h9sKW5zSvjsYNugvxJ-
TrVaEgt_DCUT0bmKhQScRrjMB6P9O50uznPwT66oF1NnU_G0HVhRzS5voiXE-y7tT3z0M3-
8A6YK009Bk_digVUul-HA4Fpd5IjoBBGe3yzaQ2ZvzarvRuipvNEQCI0onpfuZZJQ0d8
```

Signed Statement

```
{
    "version": "1.0.0",
    "id": "33cff416-e331-4c9d-969e-5373a1756120",
    "actor": {
        "mbox": "mailto:example@example.com",
        "name": "Example Learner",
        "objectType": "Agent"
    },
    "verb": {
        "id": "http://adlnet.gov/expapi/verbs/experienced",
        "display": {
            "en-US": "experienced"
        }
    },
    "object": {
        "id": "https://www.youtube.com/watch?v=xh4kIiH3Sm8",
        "objectType": "Activity",
        "definition": {
            "name": {
                "en-US": "Tax Tips & Information : How to File a Tax Return "
            },
            "description": {
                "en-US": "Filing a tax return will require filling out either
a 1040, 1040A or 1040EZ form"
            }
        }
    },
    "timestamp": "2013-04-01T12:00:00Z",
    "attachments": [
        {
            "usageType": "http://adlnet.gov/expapi/attachments/signature",
            "display": { "en-US": "Signature" },
            "description": { "en-US": "A test signature" },
            "contentType": "application/octet-stream",
            "length": 4235,
            "sha2": 
"672fa5fa658017f1b72d65036f13379c6ab05d4ab3b6664908d8acf0b6a0c634"
        }
    ]
}
```

Note: Attached signature not shown, see Attachments for Attachment message format.