Further thoughts on xAPI: cmi5 and beyond

Now that we have an LRS, a library to handle xAPI statement generation, and upload to the LRS let’s look at some further thoughts on xAPI, cmi5, when to use it, and when not to use it.

The first thing we need to do is figure out when is xAPI the best solution for our project. Some of the questions I would ask to help through this process:

- Are we reusing existing content or are we creating brand new material?
- Is all the content in an LMS or are we using content from external sources?
- Are we using SCORM to track our content?
- Do we need to convert courses to cmi5?
- How will we authenticate users to provide information about the agent?

There are probably many more questions, but I think these are good starting points.

Creating our own verbs and actions

One of the things that have been hard to wrap my head around is that the LMS is not the only input we can use for xAPI. As long as the data produces valid statements to feed the LRS, and the request is properly authenticated then it doesn’t really matter where it comes from.

We can provide a tailored experience beyond a basic set provided in Activity Streams and the xAPI specification.

I have a subdomain reserved for XML namespaces and namespace-related work so that may be a good starting point.

The solution seems to be creating your own xAPI profiles for your data and then making them available for others when they use your verbs and activities.

By defining one or more profiles we can define these specific verbs and activities that are relevant to us and that, when people outside my project or company read the data, will make sense to them.

It will also allow for reuse. If people understand what I meant when I said, person A implemented an API on project x and they agree with it, they can also use my implementation of the verb on their own projects.

The one thing I see as a drawback is that xAPI profiles are written using linked data and linked data is not the easiest technology to learn.

For examples of xAPI profiles see the xapi-authored-profiles Github Repository

For more information on Linked Data see JSON for Linking Data, Understand how structured data works, and Generate structured data with custom JavaScript

LRS: One or many? What to use them for?

The best analogy I've heard about the LRS is to think of it as a database where you store xAPI statements.

If we keep this LRS-as-database metaphor then it stands to reason that we could have multiple LRS running in different parts of the organizations. For example, we could have an LRS tracking information about the support organization and what kinds of tickets they work through while another LRS could track information about the engineering team.

Both of these LRS instances could forward information to a master LRS sitting in HR so that the HR organization can get a holistic view of performance and learning throughout statement forwarding

The other option is to have a single LRS that all clients send their data to regardless of the function and the type of statements they expect. This would make it easier to implement but it requires careful thinking about what data we're trying to gather and what additional processing we want to do with it.

Taking a deeper look at cmi5

Think of cmi5 as the spiritual successor to SCORM. cmi5 is a "profile" for using the xAPI specification with traditional learning management (LMS) systems. Unlike SCORM, it provides a set of interoperability guidelines for making courses work across compliant systems. Some of the guidelines include:

- Content Launch Mechanism
- Authentication
- Session Management
- Reporting
- Course Structure

cmi5 uses xAPI as the communication vehicle for having the courses talk to an LRS.

Turning our content into a cmi5-compliant package

There are libraries we can use to make our content cmi5 compliant

I will use the cmi5-Client-Library and its example as the basis for the content I want to build. The listing below shows the content of the example course.

```
Examples
├── AUExample1.html
├── Content
|   ├── bootstrap-theme.css
|   ├── bootstrap-theme.css.map
|   ├── bootstrap-theme.min.css
|   ├── bootstrap.css
|   ├── bootstrap.css.map
|   ├── bootstrap.min.css
|   └── prettify.min.css
├── Scripts
|   ├── cmi5Controller.js
|   ├── cmi5Wrapper.js
|   └── xapiwrapper.min.js
└── cmi5.xml
```

We'll concentrate on three files:

- cmi5.xml
- AUExample1.html
- cmi5Controller.js

cmi5.xml

cmi5.xml tells the LMS how to work with the Assignable Units, the content inside the package, and how they interact together. If you've done work with SCORM, this is similar to the imsmanifest.xml configuration file.

AUExample1.html

AUExample1.html holds the learning content and instructions for how to communicate information about the content to the LRS using xAPI. It references jQuery, Bootstrap CSS (for the UI) and xAPI and CMI libraries written in Javascript to perform the actual communication.

cmi5Controller.js

cmi5Controller.js is one of the three files that manage communication between the AU, the LMS and the LRS, the other two files are cmi5Wrapper.js and xapiwrapper.min.js

Future ideas

A future iteration of this project will try the following:

- Remove Bootstrap and replace it with either Foundation or Material Design
    - This is a matter of preference and doesn't imply a value judgment on Bootstrap
- Move to TypeScript as the primary language
- Adopt front end best practices
- Move from XHR to the fetch API
- Create a Gulp-based build system
- Remove jQuery as a dependency
    - If removal is not an option then update it to the current version (3.5.1 when writing this post)
    - If removal is not an option also evaluate if the slim version of the library would work better. The slim package excludes the ajax and effects modules so it provides a smaller download
- Bundle the scripts using Rollup or Webpack
    - This is particularly important when working with large AUs, a large number of AUs or AUs that use a large number of libraries

Begin with the end in mind

When designing instructional content we need to go back and take a look at our instructional design strategies and whatever work we've already done in developing content. I'm old school and still use the ADDIE model when working with instructional content. If this is not your first time building content in an xAPI/CMI5 environment you can look at your content through an xAPI-enhanced ADDIE methodology.

ADDIE Step	How xAPI helps
Analysis	xAPI gives us a larger dataset for analysis, making it easier to identify performance gaps, learner needs, and to tailor new learning experiences to meet those needs. Leverage data already in the LRS for the analysis.
Design	The hardest part for me to wrap my head around was that we're no longer limited to LMS data. We need to consider what we will track and what data we will capture from each source. Consider how each element will integrate into the overall learning experience. Consider the technologies to be used; you are not longer limited to what we can put inside a SCORM course
Development	Make use of code libraries and other resources during development. Take advantage of existing xAPI profiles
Implementation	cmi5 content becomes platform-independent. We can leverage existing resources and we can use modern technologies in our learning content.
Evaluation	xAPI enables evaluation at all four of Kirkpatrick's levels of evaluation including the impact on learner's behavior and the organization's performance. Because we store the results of the learning experience outside the LMS we can do longitudinal studies of our training effectiveness.

But it's important that we don't lose sight of our objective. It is not about the technology itself but how we can best use the technology to enhance the learning experience.

Links and resources

- Cmi5
  - cmi5: Technical 101
  - Incorporating a TinCan LRS into an LMS
- xAPI spec
  - Part One: About the Experience API
  - Part Two: Experience API Data
  - Part Three: Data Processing, Validation, and Security
- xAPI profiles
  - Part One: About xAPI Profiles
  - Part Two: xAPI Profiles Structure
  - Part Three: xAPI Profiles Communication and Processing
  - xAPI Vocab Server
  - Authoring Profiles
  - Profile Recipes vs. xAPI Profiles
- LRS
  - Learning Locker
  - Rustici SCORM Cloud
- Evaluation
  - Leveraging Interoperable Data to Improve Training Effectiveness Using the Experience API (xAPI)