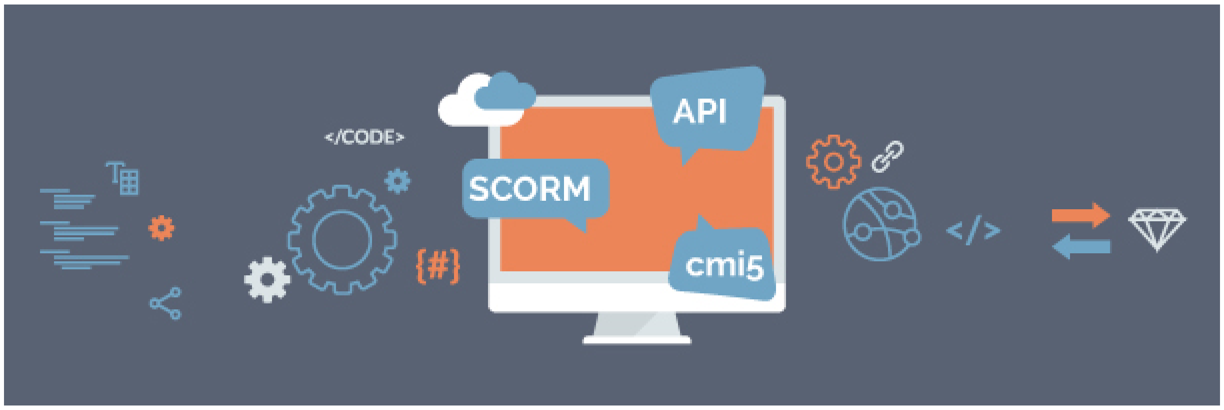
Experience API, cmi5, and Future SCORM (May 15)



by Art Werkenthin

May 21, 2015

“So can you use cmi5 today instead of SCORM? Not quite yet. The cmi5 specification has just been released as a version called ‘Sandstone.’ This is basically a stable beta, meaning you can develop against the specification and ADL promises not to make major changes. The goal is to release a final version of the specification in September 2015.”

The current standard for eLearning-to-Learning Management System (LMS) communication is SCORM, the Shareable Content Object Reference Model. Originally released in 2001, SCORM is rapidly becoming obsolete. With the successful roll out of the Experience API (xAPI), the foundation for a new SCORM standard has been laid. My purpose in this article is to outline the path to that new standard.

The history of LMS-to-content communication goes back over twenty years, to 1993. That was the year that the Aviation Industry Computer-Based Training Committee (AICC) released “CMI Guidelines for Interoperability,” originally a file-based specification for the desktop, later revisions were designed for a web browser.

SCORM itself was first released by Advanced Distributed Learning (ADL), a division of the Department of Defense, in 2001, with a major revision in 2004. Easily the most widely adopted specification for LMS-to-content communication, SCORM was originally based on AICC.

xAPI was released in 2013. It is the first component of ADL’s Training & Learning Architecture, however, it is not a specification for LMS-to-content communication. By itself, xAPI is not and was never intended as a replacement for SCORM.

xAPI defines communication between a learning experience and the Learning Record Store, or LRS. While most of us agree that the majority of learning occurs outside the LMS, there is still some formal eLearning that will be maintained in the LMS, so a more modern SCORM is certainly needed. Now that ADL has taken over the cmi5 specification, it is clear that cmi5 is the next generation of SCORM.

SCORM: The good and the bad

SCORM provides:

- **LMS-to-content communication**
  SCORM defines a way for the LMS and training content to communicate with each other. It defines both the communication layer and the data that will be stored. That data is fixed: you cannot store parameters not defined by SCORM.
- **Designed for the desktop and browser**
  SCORM was designed for browser-based communication. Mobile devices are not considered in SCORM because it was released in 2004, three years before smartphones were in widespread use and before tablets.
- **Interoperability**
  This is a key advantage of SCORM. SCORM uses a common packaging, communication, and launching mechanism. A course designer can build an eLearning module, package it as SCORM, and any LMS that supports SCORM should be able import, launch, and track the module.

ADL, recognizing that SCORM needed updating, commissioned a study called Project Tin Can. The study findings are on ADL's website, but here are some highlights:

- **With SCORM, content must reside in the same domain as the LMS**
  This is very inefficient, especially in our new "cloud-based" world. To be sure, there are workarounds for this problem, but they are not part of the specification and they are not interoperable across systems.
- **SCORM is complicated**
  The SCORM specification is long and complicated. It takes dedicated resources to develop and maintain compliance.
- **Designed for the web browser**
  A modern specification should work on mobile devices and should support mobile apps. Simulations and gaming are now in widespread use for eLearning but SCORM does not support them.
- **SCORM is easily hacked**
  SCORM uses old technology that is quite easily hacked.

Doesn't xAPI fix all that?

The xAPI is just the first component of the Training & Learning Architecture and does not replace everything SCORM does. At its core, xAPI is a data transport and storage mechanism.

Figure 1 shows some of the things your LMS + SCORM system provides.

xAPI could replace the record storage, but what about all the other functions? xAPI is not designed for scheduling, for example. There's no sequencing, no user management features, and so on.

I can hear you saying, "But 70 percent of all learning is social and xAPI is perfect for that." Yes, xAPI is great for tracking social learning, but most organizations simply cannot eliminate formal learning.

Let me give you an example: As I write this, I'm getting ready to get on an airplane. I want to get on one where the team that attached the engines were formally trained on how to attach engines to aircraft. I want them to attach the engines based on a specification designed by a formally trained engineer and trained with materials developed by an instructional designer. Social learning is great, but it is not the solution for every situation.

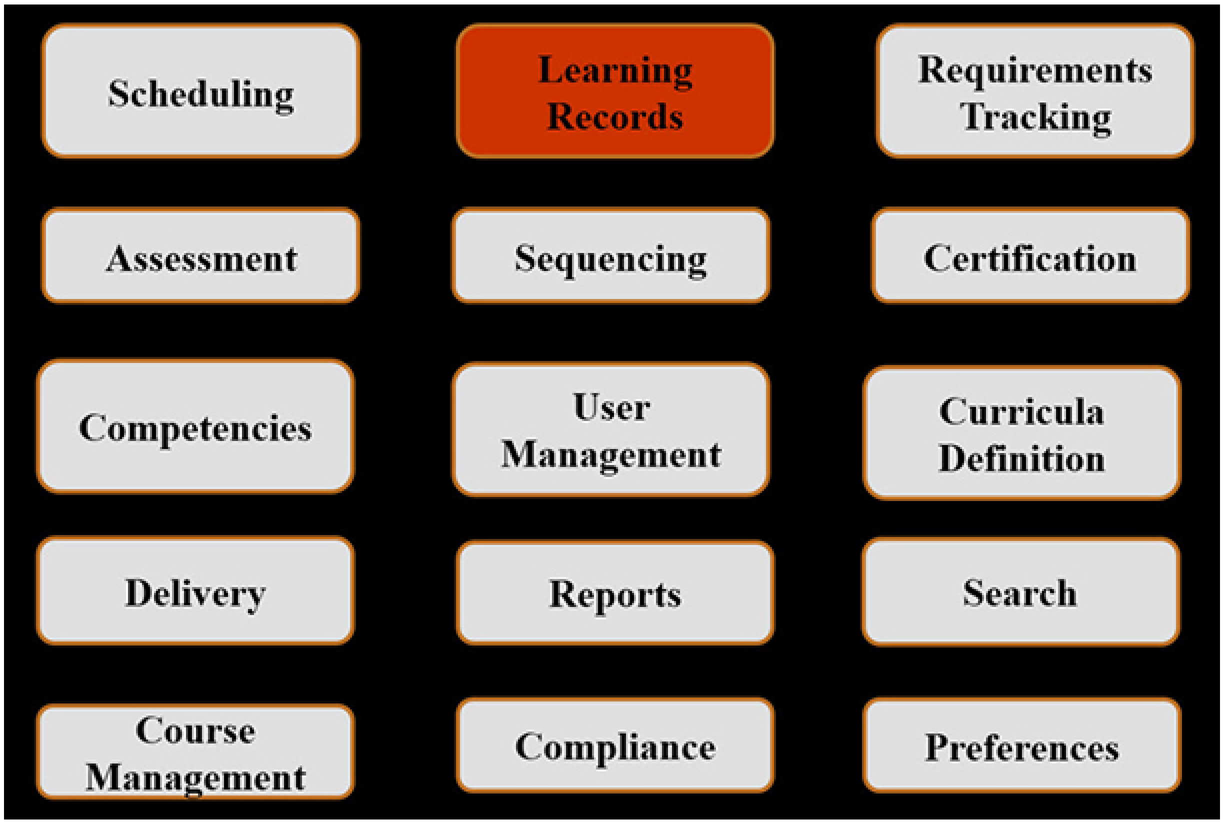


Figure 1: **LMS + SCORM provides these functions. xAPI can only handle one of them**

xAPI is flexible. Is it too flexible?

One of the great things about xAPI is that you can define your own verbs and extensions. This allows you to track everything needed to analyze your learner's experience, but it is another reason that xAPI is not the "new SCORM." For example, what verbs or extensions indicate "completion" of a course? There's no definition. Sure, you can make your own choice, but interoperability is lost. There's also no common way to define sequencing and bookmarking criteria, two basic tenets of SCORM. If we want to use xAPI as the basis of a new SCORM, we need to add a few rules to insure consistency across systems.

Introducing cmi5: xAPI with rules

Recognizing the problems with both the SCORM and AICC specifications, and to tap the new benefits of xAPI, in 2012 the AICC and ADL began working on a new specification for LMS-to-Assignable Unit (AU) communication. This new spec, dubbed cmi5, would use the xAPI as the communication and data layer, combine the features of both the AICC and SCORM specifications, and tap the new benefits of xAPI.

You can think of cmi5 as the LMS use case for xAPI. The cmi5 specification defines how the LMS and the content will communicate using the LRS.

cmi5 goals

ADL developed cmi5 with the following goals:

- **Interoperability**
  A cmi5 assignable unit should work the same across all LMS systems that support the specification. Think of the SCORM package, where an LMS imports a SCORM course. cmi5 has a similar import specification, but with cmi5 only the course structure is imported, not the actual content. This means the content can reside anywhere—behind a firewall, as an app on a mobile device, etc.
- **Extensibility**
  Unlike SCORM, the data cmi5 tracks is not limited. Since cmi5 is based on xAPI it supports extensions. You can also track binary data like videos, pictures, and audio clips. You can even share data across multiple assignable units.
- **Mobile Support**
  Here again, cmi5 benefits from xAPI: since the base communication mechanism handles mobile devices, so does cmi5.

Sample cmi5 rule

So if cmi5 is "xAPI with rules," what does a rule look like? These are the cmi5 verbs:

- **Launched**
  A "Launched" statement indicates that the LMS has launched the AU. It should be used in combination with the "Initialized" statement sent by the AU in a reasonable period of time to determine whether the AU was successfully launched.
- **Initialized**
  The AU uses an "Initialized" statement to indicate that it has been fully started and is ready for student interaction. It must follow the "Launched" statement created by the LMS within a reasonable period of time.
- **Completed**
  The AU records the "Completed" statement when the learner has experienced all relevant material in the AU.
- **Passed**
  The AU issues the "Passed" statement when the learner has attempted and successfully passed the judged activity in the AU.
- **Failed**
  The AU records a "Failed" statement when the learner has attempted and failed the judged activity in the AU.
- **Abandoned**
  The LMS uses the "Terminated" statement to determine that the AU session has ended. In the absence of a "Terminated" statement the LMS will make the determination if an AU abnormally terminated a session by monitoring new statements or State API calls made for the same learner/course registration for a different AU. When abnormal termination is detected, the LMS writes an "Abandoned" statement.
- **Waived**
  A "Waived" statement is used by the LMS to indicate that the AU may be skipped by the Learner. The LMS makes this determination based on the course structure in cmi5.
- **Terminated**
  The AU must record a statement containing the "Terminated" verb as the last statement recorded by the AU in a session.

- **Satisfied**
  The LMS writes a “Satisfied” statement when the learner has met the “move on” criteria for all AUs in a block or all AUs in a course.

So cmi5 defines some verbs that must be used according to rules in the specification, but cmi5 developers are not limited to these verbs. You can record any statements you wish in the LRS, without impacting cmi5 compliance.

Other cmi5 features

Here are some other cool cmi5 features:

1. **Launch Mechanisms**
   The course structure defines whether the AU can be launched in the existing window (where the LMS is running), or in its own window.
2. **Content Entitlement**
   There are two types of content entitlement supported by cmi5. The first is an entitlement key in the course structure. Recognizing that some developers may use a pay-per-use model, cmi5 also allows for a separate entitlement key to be determined at runtime, using a mechanism agreed upon by the LMS and the AU.
3. **Completion Criteria**
   Here cmi5 addresses the age-old debate “Does completed mean passed?” In cmi5, the course developer or the LMS Administrator can define a “move on” criteria. This can be “Passed,” “Completed,” or “Completed AND Passed.”
4. **AU-specific launch parameters**
   In the course structure, the AU can define any specific custom parameters that it requires.

Benefits of cmi5

Why use cmi5 over SCORM? Let’s take a look at some of the benefits:

- Content-defined data can be stored in the LRS using cmi5
- Data sharing across content
- Content-defined launch mechanism
- Distributed content

For each of these benefits, here is a sample use-case.

Use case 1: Content-defined data

You build a content module that wants to record the exact steps a user took to perform a procedure, with video.

Without cmi5

You can either customize your LMS to store the video received from the AU and customize the AU to send data to your specific LMS, or the AU can store the video in some external location.

With cmi5

All the data, including the video file, can be stored in the LRS.

Use case 2: Data sharing

You have a multi-AU course. You need data entered by the student in AU #1 displayed or used in AU #3

Without cmi5

There is no mechanism in SCORM to allow this. So you are again faced with customizing the LMS and the AUs, or the AUs can find a custom way to share data.

With cmi5

Your content can record any data it wants in the LRS. So AU #1 records the data in the LRS, and AU #3 fetches that data from the LRS.

Use case 3: Distributed content

You have a giant eLearning module with video, voice, animations, etc. that you need to deliver to students all over the world.

Without cmi5

With SCORM, your content is loaded to your LMS server in Houston and your students have a slow, agonizing user experience.

With cmi5

Your content is distributed globally through a content distribution network and your students are happy.

Mobile support

In addition to the benefits above, you also gain mobile support with cmi5. Since it is based on xAPI, cmi5 uses modern technology like REST and JSON.

cmi5 status

So can you use cmi5 today instead of SCORM? Not quite yet. The cmi5 specification has just been released as a version called “Sandstone.” This is basically a stable beta, meaning you can develop against the specification and ADL promises not to make major changes. The goal is to release a final version of the specification in September 2015.

The ADL cmi5 committee expects that the Sandstone release will encourage content development tools and LMS vendors to move forward with spec adoption.