# Implementing an Adaptive Learning System with the Use of Experience API

Koralia Papadokostaki, Spyros Panagiotakis(✉), Kostas Vassilakis, and Athanasios Malamos

Department of Informatics Engineering,
Technological Educational Institute of Crete, Heraklion, Crete, Greece
mtpl30@edu.teicrete.gr,
{spanag,amalamos}@ie.teicrete.gr, kostas@teicrete.gr

**Abstract.** With the evolution of e-learning and its transformation into mobile learning, SCORM fails to keep up with learner's need to discover knowledge through multiple and diverse sources. ADL's Experience API (xAPI) fills this gap and offers a novel and flexible way to keep track of a learner's activities and progress. In this paper, the xAPI and the concept behind it are shortly discussed, a brief comparison with SCORM is attempted and an innovative implementation of an adaptive LMS-free learning system with the use of xAPI is presented.

**Keywords:** Experience API · Tin Can API · xAPI · LRS · SCORM
ADL · Activities tracking · ADL's LRS · Adaptive learning

## 1 Introduction

As mobile devices gain constantly popularity and conquer our everyday lives and habits, online learning has shifted from traditional LMSs (Learning Management Systems) to an everyday and everywhere process [1, 2]. Games, augmented reality applications, virtual worlds, streaming platforms and social networks may nowadays serve as training or teaching sources, while mobile equipment of different types plays the role of the medium. This new mobile learning model offers surplus value to a user's learning environment [2, 3].

Identifying the need to support mobile and non-traditional sources of learning and to host tracking information for each user's learning curve, ADL (Advanced Distributed Learning) [4] has developed a new specification, the Experience API [1, 5], which is also known as xAPI. In a few words, xAPI is a "platform and content agnostic" [5] tool that can dynamically track and store activities from any platform or software system, as those aforementioned.

In this paper we will shortly address the concept of xAPI, explore its basic infrastructure and compare it to its predecessor, SCORM (Sharable Content Object Reference Model) [6]. Finally, we will present our novel implementation which consists of two standalone courses that seamlessly communicate without a Learning Management System making use of xAPI. To the best of our knowledge there is no LMS- independent application that enables adaptive sequencing path according to a user's previous activities and the interaction of two courses through xAPI. Our xAPI


A. L. Brooks et al. (Eds.): ArtsIT 2017/DLI 2017, LNICST 229, pp. 393–402, 2018.
https://doi.org/10.1007/978-3-319-76908-0_38

enabled application needs not the setup and configuration of an LMS and users only use their email in order to enter a personalized course. This is achieved by employing the JavaScript libraries which implement xAPI to store and track learning activities.

The rest of this paper is structured as below: in Sect. 2 the Experience API (xAPI) specification is introduced, in Sect. 3 xAPI's uses in education are presented, and in Sect. 4 our novel implementation is described. Finally, in Sect. 5, conclusions upon our implementation and plans for future work are discussed.

## 2 xAPI: Inside the Specification

### 2.1 ADL's SCORM: The Predecessor of Experience API

The ADL (Advanced Distributed Learning) Initiative is a US government program aiming to augment flexible, lifelong learning through the use of technology [4]. It is widely known for its SCORM specification [6], introduced in 2001, and revised up to the SCORM 2004 4th Edition. SCORM (Sharable Content Object Reference Model) intended to overcome the major problems of interoperability and reusability of learning content. Before SCORM was proposed, the process of tracking the learner's progress was tailor-made for each platform; if the company or foundation changed its LMS, the tracking process had to be redesigned and re-implemented. With the use of the SCORM model, the learning content is packaged into a format which can be transferred through various Learning Management Systems (LMSs) [3, 6, 7] accomplishing thus not only interoperability, but also reusability, traceability and longer lifecycle.

Although SCORM was welcomed with applauses, adopted, supported and compliant with popular LMSs and perhaps the most "widely used e-learning format" [8], rapid rise of technology caused its glory to gradually fade away. To start with, SCORM is tightly connected to the LMS ("LMS-centric" as stated in [3]) and cannot exist autonomously [2]. However, in a constantly changing world, where learning happens also beyond the LMS and through mobile devices (tablets, smartphones, smart television sets even gaming consoles), there is a need for support of informal and ubiquitous education [2], which is neatly described with the motto "Learning is happening everywhere" [1].

That was the vision Learning-Education-Training Systems Interoperability (LETSI) tried to realize in 2008, when it started investigating the requirements of the next generation of SCORM (SCORM 2.0). After lots of whitepapers and suggestions [9], ADL focused on standardized experience tracking capabilities and in 2010 a Broad Agency Announcement (BAA) project evolved: the "Experience API." Rustici Software - the company that undertook the project - renamed it to "Project Tin Can" as this term implied the two-way conversation between the company and the e-learning industry [1, 5] and today the two terms are synonymous.

### 2.2 Experience API - Understanding the Basics

The new technical specification called Experience API (also known as xAPI or Tin Can API) was launched in 2012, under the version 0.9 and up to today several versions have

been launched adding extra functionality and clarifying many issues. The current version at the time of writing is version 1.0.3 and was launched in September, 2016 [10]. The xAPI was and remains an open source, learning technologies interoperability specification that describes tracking of learner activities and experiences between technologies [11]. It is licensed under the Apache License, Version 2.0 and is widely updated and supported by the community [5].

Based on the concept of activity streams that popular social media, such as the Facebook and the Twitter, already use, xAPI can capture learning activities in the form of activity streams originating from various means and contexts [2, 7]. These records are transferred and kept in a server, called Learning Record Store (LRS), which is responsible for receiving, storing, and providing access to them. The xAPI not only specifies the structure of the streams of learning experiences, but also defines the details for their transfer and storage [7, 11]. The core elements of the specification, the (learning) activity streams are called "Statements" (xAPI statements) and describe how the learner interacted with an object, e.g. whether a learner completed a course, accomplished a quiz or watched a video. In their basic format they follow the structure of *<Actor, Verb, Object>*, but as the object can be of various types this structure can be extended by adding extra optional information, such as the result of a quiz, the timestamp of the activity or the context of an activity [11, 12]. The statements are identified by a unique UUID (Universally Unique Identifier) and are transferred and stored in JSON (JavaScript Object Notation) format.

xAPI supports a predefined Vocabulary, comprising of a large set of Verbs and Activity Types to support various cases. Verbs include *attempted, failed, experienced, shared*, while Activity Types may be *simulation, course, media, meeting, assessment* or *file* among others. Both sets are updated regularly and extended [13].

An LRS is not only a data store for statements, but it can also allow the retrieval of these statements by external applications and serves as a provider of these statements to be aggregated and analyzed. It may provide the source for data aggregation and analytics and can be the repository for extraction of precious information from basic statements [2, 7]. Apart from reporting and analytics, an LRS can be a valuable tool for personalized approaches, as activities stored in the format *<who did what>* can be easily processed. Additionally, it can host activities from various sources and that is its main advantage: whether the statement comes from a serious game, a mobile application or a webpage it can be stored under the same format in the LRS. Finally, an LRS can exchange data with other LRSs, meeting thus different requirements.

### 2.3 xAPI vs. SCORM

xAPI was advertised as the evolution of SCORM and similarities should be self-evident. Nevertheless, xAPI is a much wider technology than SCORM, can be used in various circumstances and has many advantages compared to SCORM. Firstly, in order to use SCORM, the learning contents should be delivered in SCORM packages, which can be a serious limitation for the content developer. In xAPI, though, learning activities or contents can be totally independent of data formats [14], as simple web content can be a learning activity and libraries or applications implementing the xAPI specification can provide the infrastructure for the delivery of the statements to an

LRS. This makes tracking activities from various sources a reality; statements concerning the same learner may originate from webpages, mobile applications, simulators, virtual games or social networking tools [1, 2]; all these diverse technologies can be used as training systems and data from them should end up in the same storage unit, the same LRS. The xAPI extends learning environments further than SCORM and provides independence from LMSs, fulfilling this way the vision of 'lifelong learning', since learning can happen everywhere. Additionally, as xAPI is based on the delivery of statements relative to the content and not the content itself, they are easier to implement and give the content-developers flexibility concerning the content and the hosting of the content. For example, the content may be offline and xAPI may deliver the statements to the LRS through an occasional connection to the Internet [2, 11]. This is a major advantage for xAPI, as the learner need not be constantly online, but may still contribute to the LRS with the activities that he accomplished in form of statements. Moreover, xAPI may prove to be a priceless mechanism in the hands of data analysts, as it can cooperate well with Business Analytics and reporting tools, contrary to LMSs, the traditional hosts of SCORM content [1, 14]. Finally, xAPI may simultaneously integrate with one or several LRSs, and optionally with an LMS offering this way extra value to the administrator of the data. Figure 1 illustrates the vast diversity of sources for the statements of xAPI and the flexibility in use of the data, that xAPI and its essential component, LRS, provide.

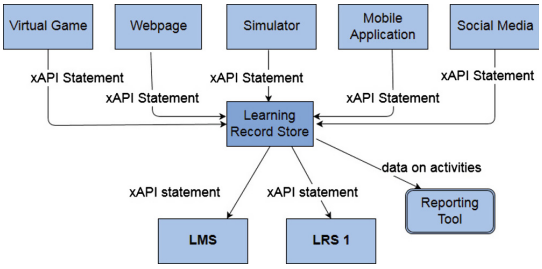


**Fig. 1.** xAPI supports a distributed architecture, where statement streams can originate from diverse sources and may be delivered to several endpoints.

## 3 xAPI in Education

xAPI broadens e-learning and its potentials, by adding tracking to various learning activities in a seamless manner. It is suitable for tracking learning activities that happen in a learning system, i.e. an LMS or an online course, but it is also ideal for recording learning activities that are not hosted in traditional learning systems. As Internet becomes the main repository of knowledge nowadays, online resources are potential sources of informal learning. In real life informal learning can happen everywhere and anytime and informal e-learning should follow these trails.

With Internet and mobile devices, YouTube videos, serious games, simulations or posts on social media can provide valuable knowledge to the learner; with the use of

xAPI and its implementations, all these learning activities can be captured and may contribute to the definition of each learner's personal profile. Till now, only knowledge that was delivered through formal e-education could be recorded, now tracking informal learning may give us additional information upon the learner, the content and the learning process. Keeping a record of an individual's learning experiences can play an important role in providing him with the proper content in the most efficient manner, which is the goal of Adaptive Learning/training [15]. Building an adaptive learning system may alter the content to meet the learner's needs or might change the way the content is presented according to the learner's profile [7, 15, 16].

From the perspective of Learning Analytics, where educational data is collected and analyzed aiming in the discovery of patterns in learning process or problems in student performance, xAPI is indeed a very promising technology [17]. In the five stages of collecting, reporting, predicting, acting and refining [17], xAPI can pioneer in collecting data from various sources (not necessarily LMSs) and provide aggregate or summarized data to third-party tools for reporting and predicting [17]. Extracting Analytics and therefore knowledge from gathered data can be used by students as self-awareness tools; by teachers for self-evaluation and detection of issues in their classroom or as a motive for improvement, while schools may use tools for their planning, decision-making and as part of Business Intelligence [17, 18].

Furthermore, xAPI can promote collaborative learning through the use of collaborative applications, social media or even serious or virtual games. Using it may augment teamwork and may convert e-learning from personal learning to team-learning [2, 19].

## 4 Implementing an Adaptive Learning System with xAPI

### 4.1 Our Implementation

Our vision was to take advantage of the capabilities of xAPI in order to create an adaptive learning system [7, 15] which will adjust its content according to the previous activities a learner has accomplished. For the learning system to be effective and accurate we had to use the online delivery of the statements from the activity provider towards the LRS. Additionally, our learning system should have access to the statements in the LRS, in order to change the content accordingly. In our case, the Activity Provider and the Activity Consumer are actually parts of the same application. The intermediate service, the LRS, stores the statements and acts as a server to our client-server application. However, the decision-making is made in the client as the course runs on the browser.

Our implementation is addressed towards fifth and sixth grade students of Primary Education. It is a brief course on spreadsheets that includes a short introduction on their use and usability and demonstrates basic concepts about sheets, cells and their format. As spreadsheets belong to the same office suite with word processors and presentations software, students may be familiar with some features of this software. Therefore, our application consists of two independent courses which are two separate webpages.

Course 1 is about text formatting and may be included in word processors, spreadsheets and presentations, whereas Course 2 provides learning content for spreadsheets.

When the learner accomplishes Course 1, a statement is sent to the LRS. When he launches Course 2, the course 'asks' the LRS if that statement exists in the LRS (Fig. 2) and if it does, it does not show the content which was included in Course 1. If the statement does not exist - i.e. the learner has not come across text formatting - the content regarding the Course 1 is displayed to the learner. For instance, if Course 1 was offered as part of a word processing lesson but the student was absent at that time or failed that course, he should revise this content and therefore our implementation in Course 2 should provide him with the information included in the Course 1. This way, our implementation offers personalized pathway to the learner according to his previous activities and adapts the content according to the learner's history and needs.

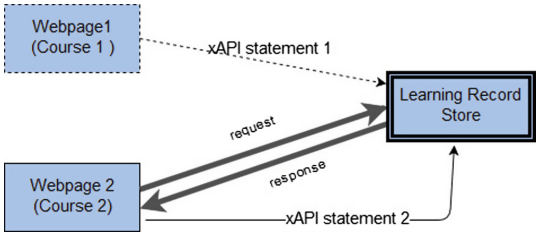


**Fig. 2.** The architecture of our implementation. A dashed box suggests that the course is not required and dashed lines that the relevant statement may not be sent to the LRS. Thick arrows show the communication between the course and the LRS regarding the existence of xAPI statement1.

### 4.2 Constructing a Course with the Use of xAPI

For our implementation, an instance of ADL's Open Source Learning Record Store (LRS) [20] was installed in an UBUNTU server. Inventors of xAPI along with the community have developed libraries in several languages, e.g. JavaScript, C, Java, PHP and Python in order to implement the xAPI specification. The library in JavaScript, called Tin Can.js, is constantly supported and updated by developers and seemed the ideal solution for us to implement our course.

The courses are webpages (.html files) with JavaScript code which performs the communication between the courses and the LRS. In order to develop user-friendly courses that would be enriched with attractive interface, multimedia content and interactive quizzes, we used a demo version of Articulate Storyline 2 [21], which is a popular software for creating learning content. Articulate Storyline 2 efficiently supports Tin Can API, but its integrated features provided one-way delivery of statements, i.e. from the course to the LRS and not vice versa, and were not sufficient in our implementation. Therefore, the Tin Can API JavaScript library [22] has been used to provide all the necessary functions making bidirectional communication between our html courses and the LRS possible. Our courses in Articulate Storyline 2 were augmented with JavaScript code calling functions to make delivery of the statements to and

from the LRS possible. As stated earlier, Course 1 is an optional part which might have been attempted in the past by a student and may belong to a different class. If it is completed successfully, our implementation sends a statement (via the JavaScript Tin Can function *sendStatement*) to the LRS with the indication that the student has completed Course 1. The Tin Can JavaScript function *sendStatement* is implemented via Restful HTTP *PUT (*or *POST)* method [11]. Figure 3 shows the statement that is sent via a PUT function towards the LRS.

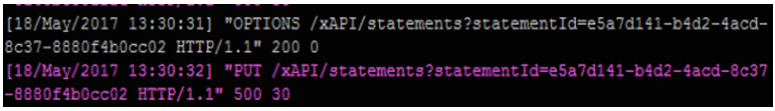


**Fig. 3.** An example of the delivery of the statement from Course 1 to the LRS.

The statement is stored in our instance of ADL's LRS (available at: http://83.212.100.157:8000/) and is shown in JSON format below.


```
{  "verb": {
  "id":
"http://adlnet.gov/expapi/verbs/completed",
        "display": {
            "und": "completed"      }    },
    "version": "1.0.2",
    "timestamp": "2017-05-18T17:30:32.680Z",
    "object": {
        "id": "http://koralia/Test_in_text_Formatting",
        "objectType": "Activity"},
    "actor": {
        "mbox": "mailto:koraliap@test.com",
        "objectType": "Agent"},
    "stored": "2017-05-18T17:30:31.282349+00:00",
    "result": {
        "completion": true,
          "score": {
            "scaled": 1 },
        "success": true    },
    "id": "e5a7d141-b4d2-4acd-8c37-8880f4b0cc02",
    "authority": {
        "mbox": "mailto:mtp130@edu.teicrete.gr",
        "name": "koralia",
        "objectType": "Agent"    }}
```

The short format of the statement is

`mailto:koraliap@test.com` (*Actor*)
`http://adlnet.gov/expapi/verbs/completed` (*Verb*)
`http://koralia/Test_in_text_Formatting` (*Object*)

and follows the structure of <*Actor, Verb, Object*> which was mentioned earlier in this paper.

When the student attempts Course 2, which might be in a posterior point in time, the student need not repeat Course 1. Therefore, our implementation sends – via JavaScript Tin Can *getStatements* function– a request towards the LRS asking whether the statement with the indication that the student has completed Course 1 exists. Again Tin Can JavaScript function *getStatements* is implemented via a Restful HTTP method (illustrated in Fig. 4), the *GET* method [11].

```
[18/May/2017 13:33:29] "OPTIONS /xAPI/statements?verb=http%3A%2F%2Fadlnet.gov%2Fexpapi%2Fverbs%2Fcompleted&activity=http%3A%2F
%2Fkoralia%2FTest_in_text_Formatting HTTP/1.1" 200 0
[18/May/2017 13:33:30] "GET /xAPI/statements?verb=http%3A%2F%2Fadlnet.gov%2Fexpapi%2Fverbs%2Fcompleted&activity=http%3A%2F%2Fk
oralia%2FTest_in_text_Formatting HTTP/1.1" 200 6375
[18/May/2017 13:33:44] "OPTIONS /xAPI/statements?statementId=37f2150b-d2bf-45aa-8e22-af0aa3e3656c HTTP/1.1" 200 0
```

**Fig. 4.** An example of the request of a statement towards the LRS.

The LRS responds and if the requested statement is found, our implementation skips this part and continues with new content (Fig. 5). If the statement is not found, our implementation shows the content of the first course as well as the new content (Fig. 6). To avoid bottlenecks and delays, the request to the LRS has been sent at the beginning of the second course and the response is stored in a local variable.

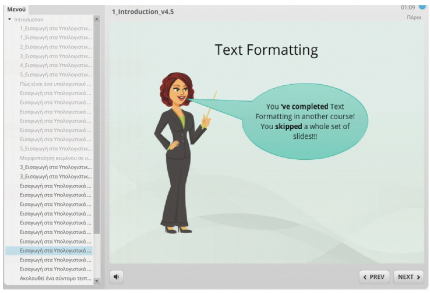


**Fig. 5.** The user has previously completed Course 1 and automatically skips the slides concerning this content. In the menu on the left, the slides concerning the content of Course 1 appear to have been skipped.

The exported courses are in html format and can be hosted in any website together or independently. They are standalone applications that need not an LMS to integrate, but instead need an LRS to communicate with. Along with the course files, the tincan.js file should be uploaded in the web servers, whereas the browser should support JavaScript.

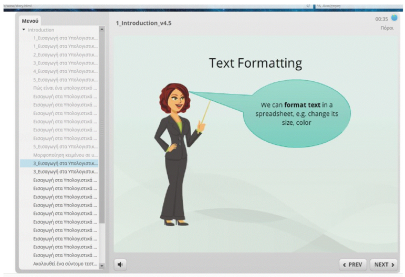


**Fig. 6.** If the user has not completed Course 1, he watches the slides concerning relevant content.

## 5 Conclusion

In this paper we have attempted a short description of xAPI, its functionality and architecture. We have compared it to its predecessor SCORM and have manifested its advantages in a constantly changing world where learning becomes ubiquitous and mobile. As activity streams gain popularity and tracking them offers valuable knowledge, xAPI is being constantly extended, updated and widely adopted in the Learning Industry [23]. Following this trend, we have created an adaptive learning system, making use of xAPI, which modifies its content according to the learner's history, without the use of an LMS.

Our implementation consists of two separate short courses illustrating the features and flexibility of xAPI. Although it was developed in a commercial e-learning authoring tool, it demonstrates the xAPI open source specification and implements the seamless communication between web content and LRS with the use of JavaScript libraries. It only requires an email address for the identification of the user and no extra authentication for him. It does not need the setup of an LMS and is totally platform independent. Additionally, the content may be scattered across various servers and platforms and each part need not coexist with the other parts, offering thus boundless potentials to the distribution of learning content.

Our implementation was evaluated by 36 students of fifth grade of a Primary School in Crete, Greece. Although the statements were massively sent to the LRS and the communication between the LRS and the course was two-way, our implementation responded with stability and accuracy; the young students noticed no delay or inconsistency during their engagement with the course. Despite the fact that the architecture of our implementation is distributed, i.e. the course and the LRS are hosted in diverse systems, the response time of our system is satisfactory (about 260 ms, where 200 ms is the roundtrip time between the user and the LRS) and does not affect the responsiveness of our implementation.

It is within our intentions to expand our application with extra functionalities, enrich it with mobile content, video streaming applications and social media integration. This way we will fully exploit the capabilities of xAPI and make a completely

personalized and adaptive course without the use of LMS. Finally, we intend to conduct Learning Analytics on statements generated by our application, hoping to form significant conclusions.

## References


1. Experience API/ Tin Can API. http://experienceapi.com
2. Murray, K., Berking, P., Haag, J., Hruska, N.: Mobile Learning and ADL' s Experience API Overview of the xAPI, pp. 45–49 (2013)
3. Lim, K.C.: Case Studies of xAPI Applications to E-Learning, pp. 11–12 (2015)
4. ADL. http://www.adlnet.gov/
5. The xAPI Overview. https://www.adlnet.gov/xAPI
6. SCORM Overview. http://www.adlnet.gov/SCORM
7. Moisa, V.: Adaptive learning management system. J. Mob. Embed. Distrib. Syst. **5**, 70–77 (2013)
8. Ruano, I., Cano, P., Gámez, J., Gómez, J.: Advanced LMS Integration of SCORM Web Laboratories, p. 4 (2016)
9. The LETSI SCORM 2.0 White Papers. https://scorm.com/tincanoverview/the-letsi-scorm-2-0-white-papers/
10. Experience API, Appendix A: Revision History. https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-About.md#Appendix1A
11. xAPI-Spec. https://github.com/adlnet/xAPI-Spec/
12. Glahn, C.: Using the ADL experience API for mobile learning, sensing, informing, encouraging, orchestrating. In: 2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST), pp. 268–273. IEEE (2013)
13. Rustici Software: Tin Can API Registry. https://registry.tincanapi.com/#home
14. Benedek, A.: Learning design versus learning experience design: is the experience API making the difference? In: Edulearn13 Proceedings, pp. 1926–1936 (2013)
15. Paramythis, A., Loidl-Reisinger, S.: Adaptive learning environments and e-learning standards. In: 2nd European Conference on e-learning, vol. 1, pp. 369–379 (2003)
16. Poeppelman, T., Hruska, M., Long, R., Amburn, C.: Interoperable performance assessment for individuals and teams using experience API. In: Proceedings of the 2nd Annual GIFT Users Symposium, pp. 1–8 (2014)
17. Del Blanco, A., Serrano, A., Freire, M., Martinez-Ortiz, I., Fernandez-Manjon, B.: E-learning standards and learning analytics. Can data collection be improved by using standard data models? In: EDUCON, 2013, pp. 1255–1261. IEEE (2013)
18. Long, P., Siemens, G.: Penetrating the fog: analytics in learning and education. In: EDUCAUSE Review, vol. 46 (2011)
19. Nine Practical Uses of the Tin Can API (xAPI). https://www.youtube.com/watch?v=8LFhDdqQ13A
20. ADL's Open Source Learning Record Store (LRS). https://github.com/adlnet/ADL_LRS
21. Articulate Storyline. https://www.articulate.com/downloads/storyline-2/
22. TinCan Javascript Library. http://rusticisoftware.github.io/TinCanJS/
23. Rustici Software: Experience API Adopters. https://experienceapi.com/adopters