

Choosing a Learning Record Store (LRS)



Advanced Distributed Learning (ADL) Co-Lab

Peter Berking

18 November 2015

Version 1.3



<http://creativecommons.org/licenses/by-nc-sa/4.0/>

Table of Contents

- 1. Purpose and scope of this paper 5**
- 2. Overview 5**
 - 2.1 What is an LRS? 5
 - 2.2 What is the xAPI? 7
 - 2.3 What problems does xAPI solve? 8
 - 2.4 How widely are LRSs used? 10
 - 2.5 Who uses LRSs and why? 10
 - 2.6 Are LRSs being subsumed by LMSs? 10
 - 2.7 What are the benefits of using an LRS? 12
 - 2.8 The importance of choosing the right LRS 12
- 3. Process for choosing an LRS 13**
- 4. Categories and examples of LRS systems 17**
 - 4.1 LRSs without data analytics engines 17
 - 4.2 LRSs with integrated data analytics engines 18
 - 4.3 LMS/LCMS with integrated LRS capability 19
 - 4.4 LMS/LCMS with API-based integration with external LRS 20
- 5. Special features and issues to consider 20**
 - 5.1 Profiles 20
 - 5.2 Controlled vocabularies 21
 - 5.3 Recipes 21
 - 5.4 Learning system integration 22
 - 5.5 Business system integration 23
 - 5.6 LRS Conformance testing 23
 - 5.7 SCORM to xAPI Roadmap 24
 - 5.8 ADL xAPI Wrapper 24
 - 5.9 ADL xAPI Lab 24
 - 5.10 ADL xAPI Statement Viewer 25
 - 5.11 Programming language and platform dependencies 25
 - 5.12 Pricing models 25
 - 5.13 Connectors 26
 - 5.14 Return on investment (ROI) 26

5.15 Open source or freeware solutions27

5.16 Learning paths and workflows29

5.17 Disconnected or occasionally connected use cases29

5.18 Security considerations for LRSs.....30

5.19 Hosting options.....31

5.20 Special requirements for U.S. DoD33

5.21 Test and staging environments.....34

5.22 Internationalization35

5.23 Enterprise LRS sharing35

5.24 The path of least resistance.....36

5.25 Aligning staff and processes to system capabilities36

5.26 Planning for operation and governance of your LRS.....37

5.27 Bandwidth to the users37

5.28 Personal data locker.....37

5.29 Multiple LRS environments.....38

5.30 Open architectures38

5.31 Component-based architecture39

5.32 Learning Experience Manager.....39

5.33 Data analytics.....40

6. Criteria for assessing quality and suitability of LRSs 42

7. For more information about LRSs and xAPI..... 51

7.1 ADL tools and resources51

7.2 Non-ADL resources54

8. References cited in this paper 57

Appendix..... 58

8.1 Sample System Requirements Matrix.....59

8.2 Sample System Features Rating Matrix.....61

NOTE: Vendor citations or descriptions in this paper are for illustrative purposes and do not constitute an endorsement by ADL. All listings of vendors and products are in alphabetical order unless otherwise noted.

1. Purpose and scope of this paper

The purpose of this paper is to help those involved in the process of choosing a learning record store (LRS) to make an informed decision. This applies to choosing an LRS for the first time, where none was already in place, and replacing an existing LRS. The paper presents a range of considerations for choosing a system; it does not contain a comprehensive survey of all available systems on the market, nor does it contain a comparative rating or evaluation of products, and should not be construed as such. For more in-depth information about systems and their features, see the references in *7 For more information about LRSs and xAPI*, consult the vendors. ADL presents this paper merely as a guide to the issues, opportunities, and processes that should be considered in choosing a system.

Because this paper is focused on LRSs, we must devote considerable attention to the Experience API (xAPI), which drives the need for an LRS, and learning data analytics, which drives the architecture, design, and features of an LRS. You must account for these in the process of choosing an LRS, since you must first determine the high-level, basic functionality you need to do the learning behavior tracking that the LRS enables.

The LRS cannot exist in isolation; to be effective, it has to be part of a larger learning ecosystem that includes learning activity providers and content that generates xAPI statements, and systems that apply data analytics, reports, and visualizations to the stored data in the LRS. This paper includes general considerations regarding this ecosystem and how the LRS functions within it; for more details about the LMS and authoring tool components of a learning ecosystem, see the ADL white papers on those topics, as follows:

- Choosing an LMS
<http://www.adlnet.gov/resources/choosing-an-lms/index.html>
- Choosing Authoring Tools
<http://www.adlnet.gov/resources/choosing-authoring-tools/index.html>

Also, despite the fact that an LRS can track offline learning behavior and even system behavior, most LRSs are predicated on tracking and reporting on learners engaged in asynchronous eLearning. To the degree that most LRSs are acquired for this purpose, we focus on that use case in this paper.

2. Overview

2.1 What is an LRS?

The xAPI spec documentation defines an LRS as “A system that stores learning information. Prior to the xAPI, most LRSs were Learning Management Systems (LMSs); however this document uses the term LRS to be clear that a full LMS is not necessary to implement the xAPI. The xAPI is dependent on an LRS to function.” (ADL, 2013).

It is important to understand that the LRS is a cloud-based service that only deals with learning information storage and retrieval of learning information (i.e., xAPI statements). It does not include the myriad functions of an LMS, thus is not a replacement for one. This “lightweight” aspect of an LRS is appealing to some; not including the overhead bulk of LMS functions significantly reduces cost and complexity. However, LRSs can be made interoperable with or even integrated into LMSs, and often are, in cases where the LMS remains as the system of record for training records. It remains to be seen whether LRSs will exist primarily as a capability built into other systems (like LMSs) rather than a separate system, but right now, they are being sold mostly as a separate system.

As a comparison between LRSs and LMSs, the following is a list of general functions normally provided by an LMS. LRS functions are highlighted:

- **Structure** – centralization and organization of all learning-related functions into one system, enabling efficient access to these functions via layered interface navigation functions.
- **Security** – protection from unauthorized access to courses, learner records, and administrative functions.
- **Registration** – finding and selecting or assigning courses, curricula, etc. by learners and their supervisors. This may include instructor-led training classes.
- **Delivery** – on-demand delivery of learning content and experiences to learners.
- **Interaction** – learner interaction with the content and communication between learners, instructors, course administrators, as well as between communicative content and the LMS (i.e. SCORM content).
- **Assessment** – administering assessments and the collection, tracking, and storing of assessment data, with further actions taken (possibly in other systems) based on the results of assessment. Many LMSs include the ability to create assessments as well.
- **Tracking** – tracking of learner data including progress on a predefined set of training goals and requirements, and tracking of courses for usage, especially in relation to required deployment of mandated training (for example, compliance training).
- **Reporting** – extraction and presentation of information by administrators and stakeholders about learners and courses, including the information that is tracked as described above.
- **Record keeping** – storage and maintenance of data about learners. This includes both demographical info profiling learners and their training progress and accomplishments. This is especially critical when an LMS is deployed as the official “system of record” for an organization.
- **Facilitating Reuse** – searching and recombining courses and possibly parts of courses for delivery in different curricula and learning tracks (this is a much more prominent feature of LCMSs, but can be included in an LMS).
- **Personalization** – configuration of LMS functions, interfaces, and features by learners and administrators to match personal preferences, organizational needs, etc.
- **Integration** – exchange of data with external systems to facilitate enterprise-wide tracking of learner performance and transfer of user data and to exploit external content and learning resources (i.e. content management systems).
- **Administration** – centralized management all of the functions in this list.

Note: Some LRSs include a Reporting function that presents xAPI data that is recorded in the LRS; however, it is not part of the core spec, and is thus not highlighted above, although it is strongly implied as an auxiliary function.

2.2 What is the xAPI?

The Advanced Distributed Learning (ADL) has termed the next generation of SCORM as the Training and Learning Architecture (TLA). All current and planned future ADL technical projects, specifications and standards efforts fall within the scope of the TLA, an umbrella term that covers projects designed to create a rich environment for connected training and learning. Phase I of the TLA has resulted in development of the xAPI, which includes learning experience tracking in these four areas:

- A new runtime API
- A new data model
- A new data model format/syntax
- A new transport/communication method

The overall TLA vision also includes concepts for learner profiles, competencies, and intelligent content brokering to meet the needs for individualized learning content and systems. The TLA is not intended to replace SCORM, but SCORM, and multiple other types of content formats, will work in the TLA. The four components of the TLA are:

- Experience tracking
- Learner profile
- Content brokering
- Competency infrastructure

The xAPI is ADL's response to the need for experience tracking; the other three components are currently under development. These other components will be developed to integrate tightly with the xAPI component. This means that legacy LRSs will need to flexibly accommodate them.

For more information on the TLA, see <http://www.adlnet.gov/introducing-the-training-and-learning-architecture-tla>.

The Advanced Distributed Learning (ADL) project to develop the xAPI grew out of a need to modernize the Sharable Content Object Reference Model (SCORM), a specification that allows courseware to be interoperable with LMSs. In 2011, Rustici Software was awarded a contract to develop the xAPI, branded as "Project Tin Can." In April 2012, ADL released the first version of the xAPI specification. The current version at the time of this paper is v.1.0.2.

It is important to understand that the xAPI augments the SCORM and does not replace it. It only (potentially) replaces the data communications protocols and models of SCORM; the other aspects of the SCORM such as content packaging and delivery are not covered by the xAPI.

The xAPI is based on the "Activity Stream" specification. The specification was a collaboration between Google, Facebook, Microsoft and other industry giants to interchange social experiences in a standard format. One key aspect of activity streams is that they are both

machine readable as well as human readable, which adds context that was never available before to both groups.

xAPI statements are written in Javascript Object Notation Language, which is similar to XML. The xAPI is an integrated approach to generate and capture learning stream data, and then organize that data into meaningful learning contexts. It is an interoperable way to encapsulate and exchange learning data through the use of a learning-based activity stream. This activity stream data includes defined actors, verbs, and activities associated with the learning experience. Below you can see some examples of learning-based activity streams that can be coded into xAPI statements.

- John Connor attempted “The War of 1812, Part 1”
- John Connor watched “The Battle of New Orleans Video”
- John Connor attempted “The War of 1812, Assessment”
- John Connor answered “Question 1” with “True”
- John Connor answered “Correctly”
- John Connor answered “Question 2” with “False”
- John Connor answered “Correctly”
- John Connor answered “Question 3” with “a”
- John Connor completed “The War of 1812, Assessment”
- John Connor scored “90%” on “The War of 1812, Assessment”
- John Connor satisfied objective “Battles of the War of 1812”
- John Connor mastered objective “The War of 1812” to level “1”
- John Connor earned “The War of 1812 – Level 1 Badge”

2.3 What problems does xAPI solve?

The following objectives and requirements, identified by the eLearning community, were the key drivers for development of the xAPI.

- **Support many content types** - Tracking user interactions within virtual immersive environments (VIEs), including games, simulations, virtual worlds. This expands the scope of “content” to include learning experiences of all kinds: real world exercises, informal learning, etc.. Tracking includes group as well individual activities.
- **Simplicity to implement** - Data model that uses a human-readable strings, using a common universal schema (JSON).
- **Portable content** - Content does not need to be delivered from an LMS, and it does not need to be rendered within a web browser.
- **Improved access to run-time data** - Tracking data is not session-dependent. It is stored in very granular form in a learning record store as human readable “subject-verb-object” strings. These can be mined and manipulated to perform complex data analytics.

- **Support offline scenarios** - Tracks interactions in mobile devices (whether connected or not). Tracking data can be generated at any time during a learning experience (e.g., a live performance) and stored locally for bulk upload when connected.

The xAPI provides a way to create flexible, semantically-defined “statements” about some user activity. These are sent and stored in an LRS. These statements can be retrieved from the LRS and put to various uses, including controlling what happens in adaptive content and learning data mining and analytics, finding negative and positive correlations, and discovering what experts and novices do differently. xAPI statements have many optional fields that can be used to define characteristics of the context, activity, and user.

Here are some examples of learning experience scenarios that are possible with the xAPI. Not that you couldn't do these before, but the xAPI allows you to do them in an interoperable way that is easy to implement. These are not possible in SCORM without significant hacks.

- Track interactions with a video, including such things as:
 - Length of time learner spent watching the video
 - At what point in the video did the learner pause it, if any, and for how long.
 - How many times did the learner replay the video, and which parts.
- Allow learners to evaluate/rate each other's learning products, including such things as:
 - What learner x's rating was
 - What learner x's comment was
 - Average rating across all learners
 - Compiled comments
- Track learning progress of informal learning experiences
- Who the learner talked to
- Text exchanges
- How frequently the learner talked to them
- How long were the sessions
- Change aspects of the learning experience based on the physical location of the learner
- Which geofence areas learner entered
- How frequently the learner entered them
- How long did they stay in them

For a fuller treatment of the xAPI (which in some cases includes extensive descriptions of LRSs), see the extensive list of resources provided in [7 For more information about LRSs and xAPI](#)

2.4 How widely are LRSs used?

There are no reliable published figures currently on LRS adoption, since it is so new, and rapidly expanding. One indicator of the adoption of LRSs is inferred by unpublished results from an ELearning Guild survey that the author heard at the DevLearn 2015 conference, reporting that 53% of respondents are looking for an LRS included in their next LMS.

Lists of adopters are provided at the links below. This can provide a flavor of usage proliferation and patterns:

- ADL
<http://www.adlnet.gov/tla/experience-api/adopters>
- Rustici Software
<http://tincanapi.com/adopters/>

2.5 Who uses LRSs and why?

Information on LRS use is still preliminary and volatile, like information in *2.4 How widely are LRSs used?* However, some vendors are publishing case studies on their web sites. These include:

- ADL
<http://www.adlnet.gov/tla/experience-api/adopters.html> [some of the listed entities are not end users, but vendors who have incorporated LRSs and xAPI into their products]
- Watershed LRS
<http://watershedlrs.com/stories/>
- Wax LRS
<http://blog.saltbox.com/blog/2013/08/22/use-case-simulations-with-zebrazapps-and-wax-lrs/>
- Yet Analytics
<http://www.yetanalytics.com/blog/2015/2/26/an-xapi-use-case-analyzing-github-issue-data-in-a-training-context>
- Learning Locker
<http://ht2.co.uk/case-studies/>
- Riptide
<http://learning.riptidesoftware.com/company/case-studies/>

2.6 Are LRSs being subsumed by LMSs?

LRS capability is a logical addition to LMSs, along with adding the reporting and analytics capabilities that account for and manipulate xAPI statement data. The value proposition of adding the ability to measure micro-level learning behaviors from a variety of sources is compelling to any training manager who is an LMS customer. These behaviors could include a learner's participation in forums, how many times they have created and shared annotations to a document, and what parts of a video they reviewed more than once. ADL predicts that the prospect of the availability of these measurable micro-level training metrics will eventually

convince training and other managers to put LRS capabilities into their list of requirements not just for an LMS, but corporate portals, ERP systems, and other platforms.

However, currently, most LMSs are focused on offering the option of performing SCORM-like functionality using the xAPI instead of SCORM, rather than leveraging the unique features of the xAPI. In order to achieve the latter, vendors will need to profoundly rethink their LMS product model. There are three main areas of LRS capabilities that LMS vendors need to consider in this new product model.

One relates to the LRS's ability to track learning experiences within content other than standard eLearning, such as mobile "learnlets," simulations, and games. And it needs to track it whether or not it is launched from the parent LMS.

Second, the LMS needs to leverage the fact that an added LRS can track different kinds of data than is possible using SCORM or proprietary LMS tracking capabilities. This includes such things as:

- Attempts, levels achieved, and other milestones rather than simply complete/incomplete or test scores
- Complex learner behaviors that are not part of formal assessments
- Data from learning activities conducted by groups of learners

Third, the LMS needs to account for the fact that an LRS can track, analyze, and report on a wide range of administrative data other than learner performance, regarding such things as how content is being used (including content outside of the LMS), apparent gaps in topics and areas of knowledge, trends in learner performance, etc.

These three dimensions are particularly apparent in regards to informal learning that does not necessarily originate from the LMS. Up to now, LMSs have controlled the learning space by forcing administrators to pre-define and pre-register learning experiences in the LMS. Now, the LMS can routinely receive data (through the xAPI) of which it has not been made aware. This gets inherently tricky in terms of differentiating and giving credit for worthwhile learning experiences as opposed to meaningless ones. It also may need to account for learners (and even systems) it has never encountered before, i.e., that are not pre-registered in the LMS.

One of the issues in integrating an LRS into an LMS is that, in doing this integration, presumably the LRS inherits (and is limited to) whatever system integration capability is part of the LMS. In other words, the LRS may be more interoperable as a standalone system (that has data interfaces with the LMS) than it is as a subsystem in an LMS. The essential question here is: how easy is it to push or pull data from an external vs an internal LRS?

One of the main hurdles to tracking informal learning is incentivizing learners to manually report or attest to their or others' learning experiences outside of the LMS—or outside of any content that is instrumented to communicate xAPI statements. A user-friendly dashboard that allows administrators, instructors, and students to do this will only go so far in facilitating this; there needs to be a reason and real incentive structured in the learning environment for learners to take the trouble to report informal learning experiences.

As a business model, this probably means that LMSs need to choose between integrating an LRS and therefore accepting xAPI statements from all quarters of the enterprise, to serve as the

authoritative source of all learning records, or remaining as a system serving separate learning management and delivery purposes and publishing learning records to the LRS.

Some industry analysts such as Rustici Software (Rustici, 2015a) predict that third party reporting and analysis tools (based on xAPI) will become a large value space for organizations, especially in terms of specialized and niche training systems. LMSs will either need to have their own tools or integrate with specialized independent reporting tools.

2.7 What are the benefits of using an LRS?

The most important thing to understand about the benefits of using an LRS is that it is only necessary if you use the xAPI, and if you do use xAPI, it is mandatory to have an LRS.

Therefore, it is not a “nice to have” proposition to acquire an LRS: either you need one or you don’t. But vendors offer many different capabilities beyond the core function of an LRS, which is to store and retrieve xAPI statements. That is why choosing one can be a complex process.

A treatment of the benefits of an LRS is directly tied to the benefits of using the xAPI, since you cannot have one without the other. Those benefits can be summed up as:

- Recording learning experiences that are not limited to desktop browsers
- Support for a variety of content types, especially those outside of the realm of self-paced eLearning
- Easier to implement than SCORM and other eLearning standards
- Allow for offline or disconnected scenarios
- The ability to track learning that involves activities that take place across multiple platforms and devices
- Improve access to run-time data. Learners and systems can share learner performance data outside of the LMS and with other applications
- Support for team-based learning scenarios
- Doesn’t need to understand what data is going to be stored ahead of time (hence the term “semi-unstructured data” applied to xAPI)
- LMSs track the data but don’t expose it – xAPI does that

2.8 The importance of choosing the right LRS

Choosing a system to track your learners’ performance and your learning program effectiveness via xAPI statements is an important decision. Though most of these systems contain the same basic functionality (tracking and retrieving xAPI statements), they are optimized for interfacing with different types of external systems, and usually have built in analytics, reporting, and visualization functions that vary greatly in features. If your organization chooses a system that is not optimized for your needs, you could end up wasting your organization’s money and wasting time for your learners and administrators. You do not want your efforts in instrumenting your learning ecosystem with xAPI or building a data analytics infrastructure to go to waste.

Another critical factor in choosing these systems is durability. This relates to whether the system will have longevity in the marketplace such that it continues to be available and supported with periodic maintenance and upgrades. This is important, at least to account for evolutionary changes in the IT environment (both hardware and software) within which it operates. It also relates to whether the system will, in the future, easily incorporate revisions to the xAPI spec.

As with all enterprise systems, LRSs should also be chosen with consideration for extensibility, scalability, and, generally, how they will fit within the overall enterprise architecture of the organization. Extensibility considerations tend to take into account the modularity of the system and how services can be customized or increased to meet changing user needs. When thinking about scalability, the growth patterns and projections of the organization are important in evaluating whether or not an LRS can meet the potential volume demands through growth. Fit tends to consider the organization's other non-learning specific business needs and how the LRS will integrate and support other business-related systems. To this end, it is very important to involve IT department staff in all discussions from the very beginning.

Although this paper is primarily predicated on first-time acquirers of an LRS, most of this information is also applicable to those switching LRSs. There can be many drivers for this decision, but it usually comes down to cost and technology affordances.

3. Process for choosing an LRS

ADL recommends the following high-level process for choosing an LRS. **NOTE:** This selection process for an LRS may actually be subsumed in an LMS acquisition. In other words, the product you are choosing may actually be an LMS that has an LRS capability. In that case, LRS features are one more set of requirements on par with all of the other LMS requirements; you do not need to go through two separate processes for choosing an LMS vs choosing an LRS. The process outlined here is identical in either case.

1. Hold stakeholder meetings to **determine the basic feasibility of an LRS acquisition, and the how your organizational goals can be met with it.** You need to answer such questions as: What business problems do you hope to solve with it? What are the risks? What resources will it require? What new processes and business rules will it require? All of this needs to be looked at under the lens of feasibility. For instance, if new processes and business rules are required, who will create and enforce them? If you are going towards a xAPI data analytics-driven learning environment, who will determine the data needed, how it will be analyzed, and how it will feed back into the system? In these meetings, be sure to include all stakeholders for whom implementing the LRS will have direct or indirect (especially financial) consequences. This includes HR, T&D (Training and Development), CEO and senior leadership, and IT staff.
2. With stakeholders, **decide on a process and schedule** (preferably with a formal project plan) for how the LRS acquisition project will proceed, using the high-level steps outlined here, or some other process.
3. **Determine the high-level requirements** for your LRS. Ensure that you get input from all groups of potential users, not just stakeholders, and solicit input from your HR and IT departments. It is important to stick to only the critical, high-level, and highly differentiating requirements at this point. That will serve to quickly filter many unsuitable

candidates when you get to step 7 below. This may require a formal requirements definition effort, especially if you are a large enterprise with many different groups of potential users who may have different (and hard to predict) needs.

Be aware that there are many types of requirements (functional, usability, etc.), representing different points of view (users, administrators, stakeholders, etc.). See Wiegers's (2000) article at <http://processimpact.com/articles/reqtraps.html> for information on how to avoid "requirements traps" such as ambiguous or vague definitions.

If you have never used an LRS before, you may want to consider gaining experience with a simple, inexpensive, or homegrown system before you buy a major enterprise system. This could substantially help clarify your goals and requirements.

Some important general considerations that may impact your list of high-level requirements at this point include:

- Whether you will need support for compliance training. This will require robust tracking features and probably certain kinds of reports.
 - Whether you need to deliver commercial off-the-shelf (COTS) content, as opposed to content you develop yourself. In the former case, you will need to ensure that the COTS content is instrumented with xAPI.
 - Whether you will need your tracking data to focus broadly on HR and HRD (Human Resource Development) issues rather than strictly on traditional training. This especially applies if you are using the LRS to support a knowledge management or performance support environment.
 - Whether you want an "all-in-one" system that contains everything you need, or whether you already have some software functions or components in place that you do not need included in the LRS. Even if you do not have these functions already, you may be planning to accumulate them gradually outside of the LRS you purchase.
4. **Determine your budget** for purchasing the system and associated support/training contracts, as well as any customization you need that you predict that the system will not provide out of the box. If you are inclined at this point to use a free, open source LRS, this step is still relevant, since there are still costs associated with an open source product. See *5.15 Open source or freeware solutions* for more information about open source products. Your budget should ideally be not simply based on available funds, but a cost-benefit analysis of implementing the system; at the very least, the cost of the system should not exceed the true cost of not solving the training problems that you would be counting on the LRS to solve. Assigning dollar values to employee training problems are notoriously difficult, but when acquiring a large expensive system, a cost-benefit analysis may be worth it. To provide a rough idea of cost, a typical system costs \$250 - \$1000 per month for a hosted LRS solution. (See *5.12 Pricing models* for more information about pricing.), varying by the number of statements collected.
 5. **Determine the category of system you will need** (see *4 Categories and examples of LRS systems*). If there are only certain major capabilities that you really need, you may be able to save money by buying only the components or services you need. If you already have

a data analytics system, for instance, you want to consider acquiring or developing just the LRS tracking component, or vice versa, instead of an entire LRS system that includes both.

6. **Identify specific systems** that match the category of system you identified in step 5. Because these categories overlap, you may identify more than one category for consideration. You may decide at this point to develop your own product rather than purchase a COTS LRS. Note that if you are a U.S. government entity, the government acquisition process requires justifications for acquisition choices. You will need to validate or justify your decision to develop your own system (vs buy a COTS product).
7. **Develop and populate a system requirements matrix** that allows assessing the systems identified in step 6 against your requirements developed in step 3. See the *Appendix A Sample System Requirements Matrix* for a sample. If you are considering more than one category of system, you may want to complete a separate matrix for each different category of system you have identified as a requirement for your organization, since each category of system has its own distinct parameters and typical feature sets. After completing the separate matrices, you will then need to decide which category you will pursue, if you are intent on or limited to purchasing only one system.
8. **Filter the list of potential candidates**, eliminating those that do not meet your minimum requirements and/or are over your budget. It is important to focus on your core needs - use weighting in the provided selection matrix (see *Appendix A Sample System Requirements Matrix*) to establish the absolute vs “nice to have” requirements. Create and send requests for information (RFIs) or requests for proposals (RFPs), or other formal documentation to these candidates at this point, if that is required for your acquisition process. Templates for these documents are usually prescribed within corporate or government organizations. If not, you can find templates on learning technology consulting firm web sites, LRS vendor web sites, or by searching on the Web. Note that some small LRS vendors may consider lengthy, detailed RFPs onerous to respond to, thus may decline to respond.
9. **Compile a detailed, comprehensive features list** for all of the remaining candidate systems. You may want to start this list by sampling the features of one system that seems to be the most feature-rich, and add any features uncovered by your analysis of other systems as you complete the comparison process. Or, you can use part or all of the criteria mentioned in *6 Criteria for assessing quality and suitability of LRSs* as your features list. You may want to edit this list of features to only those that you care about now; however, this may be limiting since you may be unfamiliar with the usefulness of some features or they may become useful in the future.
10. **Develop a system features rating matrix** (see the *Appendix B: Sample System Features Rating Matrix* for a sample) that compares the systems filtered in step 8 using the features list developed in step 9. Complete as much of this matrix as possible from the systems’ documentation; if you need more information, ask their sales representatives for it (though beware of overblown claims—verify lofty ones independently if possible). Assign a numerical rating for each cell in the matrix, indicating degree of implementation of that feature; “0” would indicate that a particular LRS does not have that feature, and “10” indicates that it has a very robust implementation of the feature. The matrix should

weight each feature according to its importance to you, enabling a rollup score for each system.

11. Contact the top scoring vendors (three to five is a reasonable number) from the previous step and **ask for a presentation/demo**. Ask the vendor for a demonstration in your facility, running your content on their system. The vendor may want to present a canned demo of their product using PowerPoint or Flash, and that is fine as a general overview of the system's capabilities, but you should see how well the system expresses these capabilities within your IT environment using real content. You might also want to ask vendors to provide a list of three customers who would be willing to host site visits or talk to you *without* the vendor present. Some experiences you might want to ask these customers about are:

- Contract negotiations
- Customizations and turning on/off baseline features
- Implementation process
- Responsiveness and quality of support

You can also investigate blogs, reviews (often offered on professional organization sites) and other online resources to assess the quality of the vendor.

It is recommended that you consider creating use case scripts (scenarios that will demonstrate the system's ability to meet your specific needs), representing common, mission-critical tasks that an LRS user would perform. During their demonstration, the vendor performs the steps required to fulfill each use case. This is a good way to evaluate how effectively and smoothly the system maps into your use cases. You can also request that the vendor set up a sandbox for hands-on testing with the system by your administrators, instructors, and learners. LRS acquisitions can be expensive, so it is not unreasonable to ask for this.

It is important to establish a firm, contractually-binding baseline of what you would be buying "out of the box" vs what would require customization above and beyond that baseline. Some vendors may tell you that their system can meet certain requirements of yours, but what it really means is that the system has an architecture that allows integration of those features with some amount of customization, which is an additional charge. You should clarify with the vendor what constitutes "customization" (i.e., requires actual programming) vs "configuration" (i.e., changes that can be made by the system administrator without any programming and system integration).

You may be able to negotiate using the product free for a limited trial period. This can be very valuable for gathering user feedback and getting an idea of what the vendor relationship will be like.

12. **Augment the matrix** with the additional information gained from step 11, adding any impressions and notes from the vendor demos.
13. **Make your decision** based on feature comparison (including the weighting you have assigned for each feature) and experiences from the demo sessions, taking into account

TCO (total cost of ownership), including the application, training, “software assurance” (yearly cost that includes upgrades, version releases, etc.), maintenance, hardware that you will need to run it on, etc.), customer support, and any intangibles. Total Cost of Ownership (TCO) is usually a 5-7 year window for learning technologies such as an LRSs or LMS. As enterprise systems usually require a minimum server architecture and LAN support, another consideration is whether a hosted solution (see *5.19 Hosting options*) or component-based architecture solution (see *5.31 Component-based architecture*) may be right for you, if one is available from the vendor. Get someone (who may not be in your learning organization) who has negotiation skills and experience involved to negotiate such important terms as pricing and licensing.

After making your decision, be clear in internal communications what the system can and cannot do. In other words, “promise low, deliver high.” Make it clear to all of those who will use the system in your organization what new roles and responsibilities they will have to take on due to implementing the system, and get their buy-in early on. It is unrealistic and unfair for them to expect that system administrators will do everything for them. As users of the system, they should experience tangible benefits (if they don’t, you need to reevaluate your requirements). They should understand that “to get, they have to give.”

4. Categories and examples of LRS systems

This section describes the major categories of available systems. These categories are key to choosing a system, since they set the stage for allowing you to align your major requirements to the type of system you need. It is important to note that these categories are not mutually exclusive. Some systems have core elements that qualify them for two or more categories. However, in these lists, systems are assigned to one category as their primary intended use or design architecture.

Some argue that the primary differentiating categorization scheme of these systems should be whether they are geared for corporate (including government) or academic users. The differences are usually apparent in the terms used within the product, for example “curriculum” for academically-oriented products vs “training track” for corporate-oriented products. However, the authors feel that it is more meaningful to categorize systems more in terms of pure functionality rather than the market within which they operate, with the caveat that each functional category of system predominates in either the academic or corporate market.

The subsections in this section describe the categories of systems and list examples. Web sites for product examples are provided for further details on each system. Note that some systems appear in more than one category, pursuant to the explanation above, as they fulfill multiple purposes.

Note: the lists of examples are not comprehensive, nor do they represent an endorsement of particular products.

4.1 LRSs without data analytics engines

The xAPI specification describes a narrow minimum scope for LRSs: storing and retrieving xAPI statements. They can include support for data analytics, reporting, and visualization

engines, but that is not within the scope of the core specification. The tools in this category only supply this bare bones capability of storing and retrieving xAPI statements.

Examples are:

- Learning Locker [open source]
<https://github.com/LearningLocker/learninglocker>
- ADL LRS [open source – meant for testing only, not production]
https://github.com/adlnet/ADL_LRS
- Wax LRS[®] [“pure” LRS offered as an option – also can be purchased with data analytics engine]
<http://www.saltbox.com/wax-learning-record-store.html>

4.2 LRSs with integrated data analytics engines

You may be satisfied with the simple generic reports that are available from some low-cost LRSs. However, if you have a need to visualize, combine, aggregate, and manipulate data in anything but the most basic ways, you should consider an LRS that includes a data analytics engine that can be tailored to your particular needs, terminology and compliance requirements. These analytics engines can usually be customized to serve your environment, especially your set of existing business intelligence tools and the learning experiences you deliver and manage.

Data analytics engines display their output to dashboards. The value of these may be overemphasized, however, since many feel that the true value of xAPI lies in serving data into other systems (such as business intelligence). See *5.4 Learning system integration* and *5.5 Business system integration* for more information.

Examples are:

- GrassBlade[®]
<http://www.nextsoftwaresolutions.com/grassblade-lrs-experience-api/>
- Grovo[®]
<http://www.grovo.com/platform>
- Learning Environment[®]
<http://www.desire2learn.com/learningsuite/corporate/>
- RISC VTA Suite[®]
<http://risc-inc.com/vtasuiteproducts/vta-administrator/>
- Skillanalyzer[®]
<http://skillaware.com/en/skillanalyzer/>
- Watershed LRS[®]
<http://site.watershedlrs.com/>
- Wax LRS[®]
<http://www.saltbox.com/wax-learning-record-store.html>
- Xyleme[®]
<http://www.xyleme.com/product/analyze>

- Yet Core®
<http://www.yetanalytics.com/yet-core/>

4.3 LMS/LCMS with integrated LRS capability

In this category, we have included LMS products that have integrated another vendor's LRS into their product, or built their own LRS capability into their LMS. The integration is handled purely on the level of the LMS vendor (who may have purchased and integrated the LRS code from an LRS vendor), such that customers buying the LMS would not know that a 3rd party LRS is integrated into the product. The downside to this is that the customer does not have a choice as to which LRS product he/she wants to write statements to; by default, statements are written to the internal, integrated LRS.

One minor variation on the theme of this category is integration of LRS capability into learning systems that strictly handle only delivery of asynchronous elearning (and not other LMS capabilities), such as the Rustici SCORM Engine.

- 1xHive®
<http://www.brightcookie.com/products/>
- Brindleway DaC®
<http://brindleway.com/all-about-design-a-course/>
- Captivate Prime®
<http://www.adobe.com/products/captivateprime.html?promoid=7JJ16KCZ&mv=other>
- ChallengeMonitor®
<http://www.e-teach.ch/eteachServer.php>
- Elements®
<http://learning.riptidesoftware.com/products/elements-platform/>
- FeatherCap®
<http://feathercap.net/>
- Gyrus Aim®
<http://www.gyrus.com/gyrusaim/>
- In2itive®
<http://www.in2itive.co.uk/>
- Knowledge Guru® [optimized for gamification]
<http://www.theknowledgeguru.com/>
- Luminosity Reach®
<http://www.cm-group.co.uk/products/learning-management-system/>
- OnPoint Learning and Performance Suite®
<http://www.onpointdigital.com/>
- Rustici SCORM Engine® [handles only delivery of eLearning, not other LMS functions]
<http://scorm.com/scorm-solved/scorm-engine>

- Tessello Total Learning System[®] [optimized for social learning and coaching]
<http://tessello.co.uk>
- Valamis[®]
<http://valamis.arcusys.com/web/valamis>

See <http://scorm.com/wp-content/assets/tincandocs/Incorporating-a-Tin-Can-LRS-into-an-LMS.pdf> for technical info on how to integrate an LRS into an LMS.

Also see <http://scorm.com/tincanoverview/learning-record-store-vs-learning-management-system/> for a list of what an LMS can do that an LRS cannot.

4.4 LMS/LCMS with API-based integration with external LRS

In this category, we have included LMS products that have interfaces to external LRSs built in. That is, they have options to configure within their product which cloud LRS to send statements to, for content running on the LMS. The customer must purchase their own account on the external LRS.

The advantage in this case is that the customer can choose which LRS he/she wants to send statements to, and has separate configuration management control over that LRS. That (presumably hosted) LRS will be updated and managed by the LRS vendor; this may not be the case in the previous category, where the LMS vendor decides when and how to update the LRS capability within their LMS product. This could be a problem when the xAPI specification evolves.

- LearnUpon[®]
<https://www.learnupon.com/>

5. Special features and issues to consider

5.1 Profiles

xAPI Profiles are essentially business process models that include controlled vocabularies; recipes are contained within controlled vocabularies (see Figure 1 below). Profiles define such high-level items as:

- Handling launch
- Where the content is hosted
- How to extend verbs and activities
- How to define activity types
- When and how to use a particular content type.

CMI5 is an example of an xAPI Profile. For more information on CMI5, see <http://www.adlnet.org/capabilities/next-generation-scorm/cmi5.html>.

5.2 Controlled vocabularies

Controlled vocabularies are perhaps the most commonly used xAPI statement standardization entity (compared to Profiles and Recipes). They exist within Profiles and are established and shared by Communities of Practice (CoPs) to help facilitate interoperability of xAPI statements. They are normally used for xAPI Verbs, but they can also be applied to xAPI Activity Types, Attachments, and Extensions.

Controlled vocabularies attempt to restrict and standardize the items that can be used in xAPI statements, in order to establish semantic precision. They provide a basis for a common language, where the same word cannot be used to mean different things, and different words cannot be used to say the same thing.

Each CoP can define its own vocabulary that works for (ideally and theoretically) all possible scenarios in their domain. Items in controlled vocabularies can be defined as to how they can be used within that domain and what they actually mean, eliminating confusion and increasing interoperability and reusability. For instance, for a specific sales environment, an enterprise-wide taxonomy of verbs can be defined to form an xAPI sales profile that contains the verbs and extensions that reflect actions related to a sale.

Two different CoPs can use the same verb two different ways, each suited to their particular needs, as long as they are carefully disambiguated by each CoP, and an indication within xAPI statements is made of where the source vocabulary and definition is stored.

ADL has published an xAPI Verb Vocabulary at <http://xapi.vocab.pub/datasets/adl/verbs/>. A more technically optimized, ready-to-use list (i.e., a JavaScript file) is also available for programmers (<https://github.com/creighton/xAPIVerbs/blob/master/verbs.js>)

Controlled vocabularies can enhance the semantic interoperability of xAPI by encouraging the adoption of Resource Description Framework (RDF) principles for xAPI controlled vocabularies. RDF is a general method for conceptual description or modeling of information. It is a W3C family of specifications that is implemented in some web resources.

By using RDF to describe xAPI Controlled Vocabulary datasets, CoPs can increase discoverability and enable applications to easily consume metadata from multiple sources. Representing xAPI Controlled Vocabulary datasets as RDF also further enables decentralized publishing of Verbs, Activity Types, Attachments, and Extensions and can facilitate an open federated dataset search capability across applications, platforms, and CoPs.

All of the above does not necessarily impact your decision regarding which LRS to acquire; it has more to do with how you instrument learning experiences. However, in the future, LRSs could be built to be optimized for accepting and enforcing controlled vocabularies. Once CoPs institutionalize their respective controlled vocabularies, the LRS may be built to enforce them through a pop-down menu that allows you to select the controlled vocabulary.

5.3 Recipes

Like Controlled Vocabularies, Recipes represent best practices to follow when writing xAPI statements, making analytics easier. A recipe is usually more fine-grained than a Controlled Vocabulary. They define content types for purposes of using particular Controlled Vocabularies, for example, videos. Recipes may include standardizations of other aspects of xAPI statements,

and the context in which they are used. Recipes and examples of them are described at tincanapi.com/recipes.

Authoring tools are starting to appear that have built-in recipes; the content author simply selects the recipe appropriate for the content object, and conformance is then ensured during the authoring process.

The relationship between Profiles, Controlled Vocabularies, and Recipes is depicted in the Venn diagram in Figure 1:

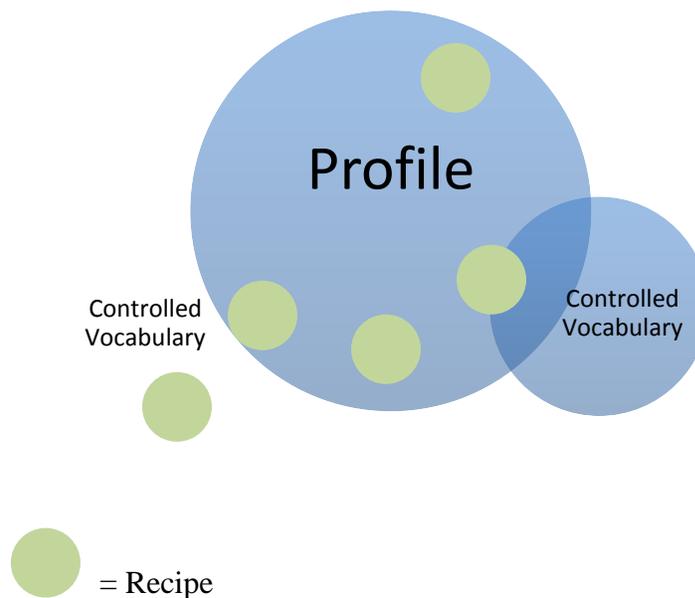


Figure 1: Relationship of Profiles, Controlled Vocabularies, and Recipes

5.4 Learning system integration

LRSs, like LMSs, need to integrate with learning support tools that are often separate but need to work in conjunction with an LRS for learning purposes. This includes general business tools that are used for learning, for instance, social media tools. The driver for learning tool integration is to facilitate control and tracking of the learning experiences recorded by the LRS.

A more difficult goal is for system integration on the level of such systems as enterprise resource planning (ERP) systems and HR systems that support and inform learning processes. The driver for this kind of integration is to avoid data redundancy and version control issues, and automate migration of data through the enterprise in a seamless process.

One particular system that is an early adopter of LRS learning system integration is video microlearning systems such as KZO Innovations. Video sharing (as short microlearning videos) is emerging quickly as a way for employees to share best practices and knowledge. Integration with these systems allows detailed information about video sharing and end user usage to be recorded in an LRS.

There are many considerations in planning for system integration. See Brandon-Hall (2012a) for a list (for LMSs, but applies to LRSs as well).

5.5 Business system integration

According to a Brandon-Hall 2011 survey (Brandon-Hall, 2012), enterprise integration is the most important requirement enterprises have for a new LMS that is to replace an existing one. But this applies to LRSs also, especially in terms of the ability of xAPI to correlate work performance data (i.e., captured in non-learning systems) with learning behavior data (capture in an LRS).

The most important business system you will need to integrate with is a business intelligence system. If you already have such a system (for instance, SAP[®] or Tableau[®]) or plan to acquire one, it is imperative that you ensure interoperability with your LRS to take full advantage of both. Business intelligence systems are more general in their purpose and design than learning analytics systems, which are referred to throughout this paper. They generally use more sophisticated statistical analysis and visualization methods than learning analytics systems, since the data is usually more complex (especially in terms of statistical capabilities), from a much wider range of sources, more technically challenging to manipulate, and more varied in its information content.

Interoperability of your LRS with business analytics systems will allow statements stored in the LRS to be used by the analytics system to produce a wide variety of employee and organizational performance indicators (not limited to learning).

In order to integrate with external business systems, some LRS such as Watershed LRS store xAPI statements in a relational database in addition to storing them in their native LRS form (non-SQL database). This takes the burden off of APIs and middleware to make this translation. External databases can then directly communicate with the LRS database.

Integration with external systems is often handled by the business system vendor; LRS vendors seldom need to get involved. This is appropriate, since the business system's value proposition is to integrate many different types of data sources, xAPI and LRSs just being one of many.

5.6 LRS Conformance testing

xAPI conformance tests currently only test the LRS, not statements sent by activity providers. Normally, an LRS will reject non-conformant content; the LRS conformance test confirms that it functions properly in this way. The LRS conformance test checks for valid JSON first, then conformance to the xAPI spec. There is no content conformance test. It is expected that a conformant LRS will record conforming statements properly and not reject them; that is essentially a way to test content.

For more information, and to download the test software, see https://github.com/adlnet/xAPI_LRS_Test.

5.7 SCORM to xAPI Roadmap

The SCORM-to-TLA Roadmap describes four phases for transitioning from SCORM to an xAPI, service-based learning platform.

This roadmap is important to consider in your LRS selection process. For instance, you may need to configure your LRS to handle LMS communication behind the scenes if you are using the “LMS as a Service” phase. LRSs may vary in the ease with which they can achieve that communication.

Figure 1 shows the phases at a high level. Full details can be found at <http://adlnet.github.io/SCORM-to-TLA-Roadmap/>

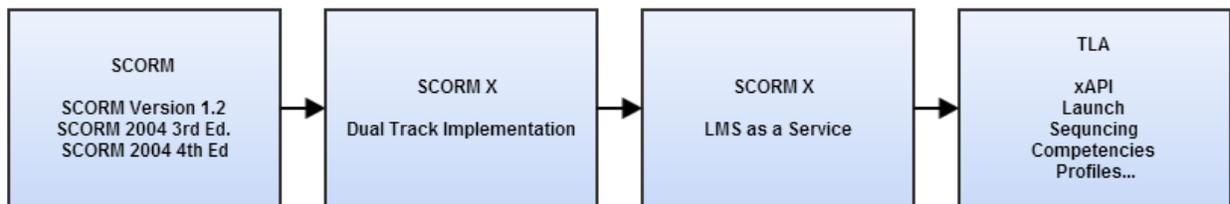


Figure x: SCORM to TLA Roadmap Phases

5.8 ADL xAPI Wrapper

The ADL xAPI Wrapper is a middleware JavaScript file that handles a lot of the technical “housekeeping” functions, making it easier to create and manage xAPI statements. It can be included in web based xAPI clients to simplify the process of connecting and communicating to an LRS. It is enclosed in an ADL object like the ADL xAPI Verbs project, allowing a single object to contain both the ADL verbs and the ADL xAPI Wrapper. For more details and to download the software, see <https://github.com/adlnet/xAPIWrapper>.

Using or not using the xAPI Wrapper has no direct impact on choosing an LRS; however, institutionalizing use of it can make it easier to maintain and troubleshoot an LRS, lowering the life cycle cost of the LRS acquisition.

5.9 ADL xAPI Lab

The ADL xAPI Lab assists in developing xAPI statements and communicating with a LRS utilizing the xAPI Wrapper (see above). It features:

- Authentication configuration
- Complex Statement Builder
- Send and Receive Statements
- JSON Validation
- Communication with Document APIs

This tool is not required to work with an LRS; however, it is recommended as an aid for non-technical users. For more details and to download the software, see <http://adlnet.github.io/xapi-lab/>.

5.10 ADL xAPI Statement Viewer

The xAPI Statement Viewer is a convenient way to view statements using data tables, and is bootstrap-based. It works in coordination with an LRS, pulling xAPI Statements from it using the xAPI Wrapper and displays them conveniently in a sortable and filterable table. It is free to use, change, and integrate into any of your projects. This is usually a built-in feature of an LRS.

See <http://adlnet.github.io/xapi-statement-viewer/> for more details and to download the software.

5.11 Programming language and platform dependencies

It is important to determine what programming language and platform dependency requirements are for an LRS you are considering purchasing, since it can have a substantial impact on cost and deployability. This also relates to customization of the product, since the programming language may be one that your programmers are not familiar with, making customization difficult (i.e., forcing you to consult with the vendor for this service).

Any programming language used to create an LRS has to be able to handle HTTP requests. This includes the REST interface.

5.12 Pricing models

As with many enterprise software systems, vendors will have their own particular pricing model that they feel positions them best in the marketplace and suits their needs; this makes it difficult to compare prices between vendors. However, there are certain basic categories of pricing models, as follows:

- **Seat-based** – this model uses the number of employees in the enterprise, or possibly the number of employees who will ever write statements to the LRS as a basis for pricing. These “seats” are a maximum number of people who may end up logging in to the system over its life cycle. There are usually tiers of seats (for example, up to 10,000 vs up to 20,000 users). Do not confuse seats as a pricing model with system capacity seats. The latter is the number of concurrent users that can safely use the system without overburdening it.

Seat-based pricing can run into problems with “extranet” users. If partners, customers, and others outside of the enterprise (i.e., other than employees) need to use the system, accounting for their numbers accurately may be complicated, and may need to be based on unreliable estimates.

- **Analyst-based** – this model is a variation on seat-based; the difference is that it uses the number of analysts who need to look at the data in the system or maintain it, rather than the number of employees who might send statements to it. These systems are sold according to numbers in each tier of users, often at least including system owners and administrators (differentiated by sets of system privileges). Normally, employees would only be sending statements to the system without an ability or need to look at the data

that has been sent (either as raw statements or reports), but employees viewing data can sometimes also be part of the equation in this analyst-based pricing model, since the LRS may give them a dashboard with a view to the data. This could be for simple feedback on their progress, or more sophisticated functions such as viewing their personal “bookmarklets” (location indicators learners place in the content which are recorded via xAPI).

- **Usage-based** – this model is based on the number of xAPI statements sent to it per month. This model is particularly attractive in the case of anticipated LRS usage surges, due to such events as new product releases and seasonal cycles. In these cases, where there may be little usage of the system except in certain short periods, paying for a baseline of seats may be less economical than paying per use, for time used.

Usage-based models can make a hosted solution especially attractive, since your organization does not need to permanently maintain a full complement of server, bandwidth, and support resources to handle the highest load times. That is the responsibility of the vendor, and you will only be charged for the (potentially) short time that usage peaks.

- **Capability-based** – this model is not based on seats or usage, but tiers of capability. For instance, the base-level product could be a “pure” LRS, without any analytics. The middle tier could include reports and external business intelligence integration. The highest tier could include single sign-on and interactive, real time analytics.

LRS integrations into an LMS product that you own or sell involve a separate pricing model. They usually involve an implementation fee, which is paid at the outset to set up the LRS to work within your LMS. After that, a license fee is charged based on the number of users or the number of statements collected by the LRS.

5.13 Connectors

If you have direct access to content, or are the activity provider, you can simply instrument it with the xAPI calls you want. Those will be sent directly to the LRS. However, if you do not have access, you can use a connector to translate data sent from the content or activity. This can be bespoke middleware, APIs, or database queries that turns program code into xAPI statements. Code libraries such as at tincanapi.com/libraries can help.

5.14 Return on investment (ROI)

It is difficult to assess an ROI for an LRS, since it works closely with other parts of the learning ecosystem, thus it is hard to isolate its value from other parts of the ecosystem. Also, learning initiatives and programs are notoriously difficult to determine an accurate ROI for. This is particular true for xAPI data applied to informal learning scenarios, where quantitative data is absent.

Your ROI calculation for acquiring an LRS to support this switch should take into account cost items such as:

- Cost of collecting and analyzing metrics currently used (i.e., non-xAPI based)

- Projections of cost of collecting new xAPI-based metrics (apart from the cost of the LRS).
- Cost of the LRS (startup and ongoing)
- Cost to instrument new and legacy learning experiences with xAPI
- The value of organizational efficiencies, i.e., performance gains, achieved through feedback gained by use of xAPI

5.15 Open source or freeware solutions

Open source options are attractive due to the absence of any licensing cost. However, it is important to be aware of the pros and cons of acquiring an open source solution, as the cost could, over the life of the system, equal or exceed a commercial system. It's easy to be enamored of the free license aspect and ignore the required (possibly extensive) customization and support that may be necessary.

It is also easy to overlook the potential advantage of open source systems in that the product can be completely tailored to the particular requirements of the organization. If managed properly, this advantage can make an open source solution cheaper, not just because the license is free, but because the development and customization efforts can be focused solely on the needs of the organization and nothing more. Contrast this with a commercial product with lots of features that your organization may not need (but you are paying for them nonetheless). The business model for a standard commercial system is to build to the widest set of possible requirements to attract the widest client base. Your organization may not have all or even most of these requirements.

All of the above being said, acquiring an open source LRS usually does save money. It is also the wave of the future in software. Some analysts have predicted that fully 50% of all software used within 5 years will be open source.

Open source systems are indicated in the lists of systems in *4 Categories and examples of LRS systems*.

On October 16, 2009, U.S. DoD issued new guidance on open source software (see <http://powdermonkey.blogs.com/files/2009oss.pdf>). The guidance establishes open source software as having equal weight as proprietary software during acquisition evaluations. It is a break from the past, when open source software was deprecated for use in DoD due to security and quality concerns. The benefits of open source software are described in this guidance document as follows (open source is referred to as OSS):

The continuous and broad peer-review enabled by publicly available source code supports software reliability and security efforts through the identification and elimination of defects that might otherwise go unrecognized by a more limited core development team.

The unrestricted ability to modify software source code enables the Department to respond more rapidly to changing situations, missions, and future threats.

Reliance on a particular software developer or vendor due to proprietary restrictions may be reduced by the use of OSS, which can be operated and maintained by multiple vendors, thus reducing barriers to entry and exit.

Since OSS typically does not have a per-seat licensing cost, it can provide a cost advantage in situations where many copies of the software may be required, and can mitigate risk of cost growth due to licensing in situations where the total number of users may not be known in advance.

Open source licenses do not restrict who can use the software or the fields of endeavor in which the software can be used. Therefore, OSS provides a net-centric licensing model that enables rapid provisioning of both known and unanticipated users.

By sharing the responsibility for maintenance of OSS with other users, the Department can benefit by reducing the total cost of ownership for software particularly compared with software for which the Department has sole responsibility for maintenance (e.g., GOTS).

OSS is particularly suitable for rapid prototyping and experimentation, where the ability to “test drive” the software with minimal costs and administrative delays can be important.

(Memorandum Clarifying Guidance Regarding Open Source Software (OSS), Oct. 16, 2009)

What is important to understand about open source software is the relationship it behooves you to build with the open source community that has arisen for the open source product you are acquiring. Staying in touch with the community in order to be able to discover and use already developed modules of functionality that you need (that are not part of the product baseline) can decrease your customization costs enormously. Open source communities often remind you that deploying open source means you are a responsible member of their community. There is an expectation that you contribute, as well as receive code, training, and documentation from the community. The cost of staying active in the community and both researching and acquiring as well as sharing your products and solutions must be factored into the level of effort for acquiring an open source tool. Open source LRSs are often backed by non-profit organizations and foundations.

It is also important to evaluate the strength and size of the open source community for the open source product you are acquiring, as well as the longevity of the product. This can mitigate obvious concerns that major sponsors of open source software can stop development at any time, or that communities can atrophy. Another possible concern is that a tool can grow so quickly in its popularity that documentation takes a back seat to development and has not caught up to the current release of the software; especially in the case of open source software, where you have no vendor who is obligated to support you, a lack of adequate documentation can make a product difficult to install, use, maintain, and troubleshoot.

Finally, the baseline versions of some open source products are very basic; some level of customization is often needed to make the software not only meet your special requirements but also meet a modest level of universally recognized functionality for the type of product. It may be risky to assume that an open source product will be usable straight out of the box. If you have no development resources ready and willing to augment the product’s functionality right after you acquire it, you may not be able to use it for some time. Some companies will eventually build their business model on selling customization services for open source LRSs, as they have done for LMSs.

Freeware may or may not also be open source. Freeware may have restrictions on copying, distributing, and making derivative works of it, where open source software does not. And freeware does not necessarily make source code available. Freeware may be restricted to

personal use, non-profit use, non-commercial use, etc. Freeware that is not open source is a risky investment, since you cannot easily customize it.

There may be special restrictions on use of freeware within your organization. For U.S. DoD, see <http://www.acq.osd.mil/ie/bei/pm/ref-library/dodd/d85001p.pdf>

5.16 Learning paths and workflows

In a move towards “LMS-izing” the LRS, some vendors have added the ability for an administrator to set up learning paths, similar to this very standard feature of LMSs (often termed in LMSs a “curriculum”). Because of the xAPI’s applicability to domains outside of learning, and because they are almost functionally identical to set up in the system, these LRSs also usually include workflows.

This feature implemented in an LRS involves defining specific verbs and context strings (forming an xAPI recipe, essentially – see 5.3 *Recipes*) to describe all of the activities in a particular learning path that needs to be created. In typical xAPI style, and unlike an LMS, this could cover offline activities not normally tracked by an LMS, and micro-level assessment behaviors. For workflows, it includes steps or tasks in a process. Statements would take the form of “User x completed item 1 (at c level of competency)”, “User x completed item 2 (at b level of competency)”, etc.

When the learner or worker is performing these activities, and statements are sent to the LRS that conform to the learning path or workflow model (i.e., xAPI recipe), achievement of items on the learning path/workflow are logged. A dashboard in the LRS shows progress towards fulfilling the path, and competency or quality scores for each milestone.

This feature is an item for probable further development by LRS vendors—internal queries of the statements in the LRS that indicate whether a sequence of events has been accomplished, that are either displayed within the LRS or sent to other systems. Completion of the sequence, possibly with a rules engine driving criteria for completion, could trigger various events within or outside of the LRS.

5.17 Disconnected or occasionally connected use cases

The ability to synch with a locally stored LRS could be an important use case for you. This concept involves allowing LRS functions to be performed in environments where there is no, intermittent, or limited bandwidth or connectivity to the cloud LRS. It often refers to mobile devices, where user network access is less stable and reliable (and can be expensive). Content that is to be consumed must be provisioned to the device at a time and place when there is stable and cheap connectivity, such as on a wireless network (as opposed to cellular data network), then used offline. The local device must have a player capability (a web browser might suffice) in order to play the content.

If there is no connectivity to the cloud LRS at runtime, the xAPI spec provides the possibility of using a locally stored LRS (i.e., contained in the app that is on the mobile device) to cache statements that would otherwise go to the cloud LRS. Later, when connectivity is achieved, the local LRS synchs with the cloud LRS. The local LRS must be built to account for this synching scenario (the cloud LRS is not aware that caching is taking place).

5.18 Security considerations for LRSs

Like any other enterprise system, LRSs must meet the security needs of the organization. This is especially true in the current era, where LRS functionality is largely delivered via the Internet, not enterprise intranets or extranets (the driver for this migration is mostly to allow greater access to learning).

For commercial installations, LRS security amounts to:

- Protecting against unauthorized login. This is primarily not so much a function of the LRS, whose login functionality relies on universal web standards, but rather the placement of the system within the corporate intranet environment and the inherent security features of that placement. Commercial entities are of course concerned about other organizations gaining competitive advantage by seeing the training of competing companies, and government has obvious security concerns, so access to the system is a primary concern.
- Locking users out of capabilities that are not included in their user profile, in other words, keeping users from writing statements to the system or viewing analytics information if they are not authorized to do so.
- For DoD organizations, there are specific considerations relating to the possible harmful effects to national security and individuals' life and limb due to unauthorized access to the system and particular courses that may be classified, etc. There are a number of issues that need to be considered in this regard. The same considerations for LMSs most likely apply to LRSs.
- Privacy policies and Personally Identifiable Information (PII) may be an issue depending on how public access to your LRS is and what kind of information you store on it about your users. EU Internet privacy rules, Canada's Freedom of Information and Protection of Privacy Act, Health Insurance Portability and Accountability Act (HIPAA), Family Educational Rights and Privacy Act (FERPA), and the US Patriot Act may be a consideration.
- It is important to find out what programming language and third party OEM components were used to build the LRSs you are considering acquiring. There are innate security considerations for some programming languages.

A number of security concerns come into play for hosted solutions, since in that case both your content and the LRS system reside outside of your firewall. These concerns generally are the same for cloud computing, which has become indispensable and ubiquitous throughout all aspects of learning technology, playing a vital role in providing the services people and employees use in their everyday life. But as cloud computing has risen in use and mission-critical importance, concerns related to privacy, data security, and even sovereignty have emerged. One partial solution is to use a "private cloud" with VPN access for those outside of the enterprise network. This however, may not work in an environment where public access to your LRS is required. Custom-designed hybrid cloud solutions are becoming more and more common to meet specialized security needs that a standard cloud cannot.

Some LRSs such as Yet Core[®] have adopted specialized security features tailored to LRSs such as:

- Immutable data
- HTTPS only
- HSTS enforced
- Reading does not equal writing – separates database from query engine to reduce threat of injection attacks

There is very little in the xAPI spec that prevents you from adding custom security features. Design features related to security that may be coming in future versions of LRSs are:

- Full stack
- Intrusion detection and alarms
- Auditing
- Zero-day responses
- Response time standards

5.19 Hosting options

There are three options for hosting most enterprise learning systems, including an LRS:

- Behind your firewall
- Vendor-hosted
- Public cloud hosted

Most LRS vendors offer the first two options; a few are now offering the last. A vendor-hosted LRS is installed and managed on the vendor's server by their staff, rather than behind your enterprise firewall by your staff (the "behind your firewall" option). Public cloud hosted solutions refer to hosting the LRS not behind your own enterprise firewall but on a public cloud service such as Amazon Web Services.

For all practical purposes, public cloud hosting is no different from hosting behind your firewall in the sense that you have full control over and management responsibilities over the LRS; the only difference is that it is hosted on rented server infrastructure outside of your firewall.

Public cloud hosting often requires a different approach than either of the other two options because the server configuration options are limited; they are controlled by the cloud service vendor. Often the vendor must make alterations to their LRS to conform to the cloud infrastructure requirements. The advantages of public cloud hosted LRS solutions are the same as with any public cloud hosted system; you do not have to acquire and maintain the server infrastructure yourself (which could be significant for a large LRS installation), and there is less load on your network. The fact of being hosted on a server outside of your firewall can raise security concerns, however (these are gradually being eased – for instance, DoD now allows Amazon Web Services hosting for some internal DoD systems). On the flip side, it may be a plus that it is not behind your firewall, if you need users outside of your enterprise network to be able to write statements to the LRS, and you don't want to worry about security breaches by outsiders coming into your network.

Vendor-hosted solutions are often termed “SaaS” (software as a service) or “cloud” solutions, although this use of terminology can be confusing. SaaS or cloud can be used to refer to a disaggregated, Internet-based collection of software services or components that make up an entire system such as an LRS. In practice, these services are almost always hosted on the vendor’s server, but they could be installed within your intranet as a private cloud or custom-designed hybrid cloud (see section *5.18 Security considerations for LRSs*).

Some of the advantages of a vendor-hosted platform are:

- Eliminates the cost of hardware and network infrastructure needed to support a local installation of the system
- Lowers your staff costs for administration and maintenance
- Puts less bandwidth load on the corporate network
- Content and feature updates can be accomplished without intervention by your staff
- Guarantees that system upgrades and patches are applied on a timely basis; most vendors upgrade their hosted installations on a monthly basis. Installation of updates on your server can lag significantly behind the vendor making them available, for a host of reasons.
- Having the vendor take responsibility for upgrades and patches avoids the headaches of reestablishing your integrations, etc.
- Enables faster implementation. This can be dramatic, for instance, 3 weeks for a vendor-hosted solution vs 6 months for a behind-the-firewall solution. In some cases, software wizards are used to simplify and step users through the process.
- Requires little or no internal technical support or development
- Provide incentives and guarantees for maintaining uptime (via financial penalties assessed against vendor).
- Provides data center compliance (esp. in regards to data centers in foreign countries) since this is handled by the vendor
- Scales more easily to account for temporary surges in usage (due to new product releases, seasonal events, etc.), due to the typical centralized system architecture usually implemented by hosting vendors, with loads dynamically shared and balanced across customer implementations.
- By virtue of the vendor taking responsibility for scaling, it eliminates the need for you to commit to purchasing and maintaining additional servers and bandwidth that may be unnecessary to support normal load during non-surge times.
- Is often associated with a usage-based pricing model (see *5.12 Pricing models*), which may be more economical
- Contractually, it can be easier to switch to another vendor or end a vendor relationship

One of the main disadvantages of a vendor-hosted solution is that it restricts opportunities and scope for local customization. Also, a vendor-hosted solution may not provide the level of security required by your organization, although vendor-hosted solutions are increasingly more secure.

The security issue relates not just to unauthorized access, but also the fact that you may be placing trade secrets and other intellectual property in xAPI statements outside of your firewall on the vendor’s server, outside of your control. If your organization’s policy prohibits this, a

vendor-hosted solution will not be right for you. And a vendor-hosted solution is summarily ruled out if there is classified data stored in the content. See section *5.18 Security considerations for LRSs* for more information on security issues.

Finally, for government entities, a vendor-hosted solution may not be an option since government rules tend to mandate outright ownership and control of systems, rather than an arrangement like a vendor-hosted solution that resembles leasing.

Most vendor-hosted solution scenarios involve a single instance of the vendor's software that is engineered to support multiple customers, rather than establishing a separate instance of the software for each customer. This enables efficiencies for the vendor whereby they can apply patches and version upgrades for many customers at the same time. This lowers the operational LOE for the vendor and allows them to focus more on developing their product. Vendor-hosted systems are vendor-maintained and managed with minimal intervention required by the customer, so much of the headache of deployment planning relating to upgrades of the software can be avoided.

Vendors who offer hosted solutions commit themselves to providing a robust hosting and networking infrastructure with uninterrupted access 24 /7 basis from any location. The system that they host must be scalable and have redundant backup and security. These are items for due diligence verification during the acquisition process, if you decide to buy a vendor-hosted solution. Guarantees of average percentage of uptime are often written into the LRS service-level contract. You may want to independently verify uptime using a Web monitoring service. These services monitor access from multiple global endpoints. If an issue arises, your mobile phone is texted. Some monitoring services are quite sophisticated. They can actually periodically read data-driven Web-page elements to validate site availability in addition to the back-end functionality.

Vendor-hosted solutions are generally more expensive (roughly 20%) because they require the vendor to assume responsibility for maintenance and administration instead of the customer.

You might want to use a “try before you buy” approach by using a vendor-hosted solution for a while before you decide to buy the system. Also, consider a vendor-hosted solution that is metered (pay-for-use price) rather than flat license for a maximum number of users.

Note: Vendor-hosted solutions are sometimes called “ASP” (application service provider) solutions. Do not confuse ASP with Active Server Pages, a web programming script.

5.20 Special requirements for U.S. DoD

The DoD Information Assurance Certification and Accreditation Process (DIACAP) is the DoD process to ensure that risk management is applied on information systems. It certifies and accredits a DoD information system to maintain the proper information assurance (IA) posture throughout the system's life cycle. If you are acquiring an LRS for a U.S. DoD organization, it is important that you check the DIACAP certification status of any LRS you seek to acquire. Also, DIACAP may require you to have your LRS hosted at the Defense Information Systems Agency (DISA) facility, not at your facility (and not hosted by the vendor, either).

You may be subject to Service-specific requirements. These requirements speak to the “fit” of the system to the enterprise architecture of the organization (in this case DoD). These cover requirements such as:

- Security
- IT environment
- Specific use case testing
- Training gap/training needs analysis capability

Each Service often has their own training records system that the LRS may need to integrate with. For instance, the Navy often requires their learning systems to integrate with NTMPS (Navy Training Management and Planning System) for personnel information and training records.

One requirement that is fairly consistent across the Services is that any LMS must interface with DEERS (Defense Enrollment Eligibility Reporting System) for user verification and registration information. This may also apply to an LRS.

- There may be particular implementation issues when installing an LRS in U.S. DoD or government, such as:
 - Requirements for conducting site or pre-installation surveys
 - Constraints on who can host the LRS
 - Hardware, software, and firewall requirements
 - Particular government contracting rules regarding setup, startup costs, vendor support, and annual maintenance agreements

5.21 Test and staging environments

It is important that you institute at least three staging environments for your LRS, possibly on three separate networks. When acquiring an LRS, you should take this into account.

Consideration of test and staging environment requirements is often an oversight until after procurement (at which point there are financial barriers to implementing it). The three environments are:

- **Development** – for activity providers to upload, configure, and test the ability of their content to write valid statements to the LRS, and for administrators to perform “what if” scenarios for major changes to the system.
- **Test** (also termed **Stage**) – for content and major configuration changes made in the Development environment to be verified and finally approved before being migrated to the Production environment. This instance of the system should exactly match the Production system in all respects.
- **Production** – The live system that learners and administrators use.

These environments do not have to be separate installations. Isolated areas or instances of one system can be just as effective; however, firewall restrictions and different access needs for the user groups associated with each of these environments may prohibit this.

Acquisition of these environments in addition to your production environment will probably affect pricing and your infrastructure requirements. Licensing can be complicated if external entities such as content development vendors need to use the additional instances. Special

licenses may be required for them. LRS vendors may in the future sell packages that include these staging environments pre-configured (“sandboxes”).

You need to be careful about allowing testing of new LRS versions/features/customization and testing content on the same environment or instance of it. This situation can lead to problems, where, for instance, content works well in the Test environment, but not in the Production system because they are not precisely the same.

5.22 Internationalization

If your learners include international audiences (especially including foreign language speakers), you will need to consider features of the LRS that will support it, as well as plan your LRS implementation accordingly. It is not just a matter of administrator interfaces being presented in the language of the administrators. There are factors related to the syntax of other languages that might come into play, at least in terms of reading xAPI statements. For instance, the subject-verb-object order may be different in another language.

There are other factors you may need to consider, in addition to language, such as:

- U.S. export laws governing dissemination of information in areas of technology that is deemed of strategic importance to national security (this applies to information that is not classified or marked as FOUO)
- Local government rules and regulations that may lead to non-compliance of content
- Accreditation differences
- Cultural norms
- Local IT environment

5.23 Enterprise LRS sharing

The driver for the popularity of instantiation of an LRS for individual business units is mostly economic; it allows an enterprise (especially a large one, like the Federal government) to save money through sharing of the same system, rather than each unit, agency, etc. having to purchase a separate system. There are two ways LRS sharing can be achieved: one is where an organization is paying for a greater capacity on an LRS than they are using and another organization can fill that capacity up to (but not over) that maximum. This can result in no extra cost to either organization except for the maintenance and administration associated with using that larger capacity. This arrangement is enabled by a seat-based pricing model (see *5.12 Pricing models*) and a license with the LRS vendor that allows the purchasing organization to share with other organizations.

The other arrangement is necessary where there is not enough unused capacity. In this case, the LRS purchaser organization buys more seats or a higher capacity tier on an LRS, with the organization using the higher capacity covering the cost. This can be highly economically advantageous for the organization using the higher capacity, since it is almost always a net savings in cost to share an existing LRS in this way rather than buy a separate product or hosting package from the vendor. This arrangement can be used when the pricing model is either seat-based or usage-based (see *5.12 Pricing models*) and is also subject to licensing rules.

5.24 The path of least resistance

It is important to remember the simple fact that most users, in many cases regardless of their skill set, will follow the path of least resistance in using an LRS, as with any other software. In other words, users will gravitate towards the most heavily optimized system features—those that are prominently available in the interface and easiest to manipulate. This can include predefined data analytics reports and visualizations; the easy availability of these can inhibit the creativity and vision to create custom objects and thus achieve a truly data-driven learning ecosystem.

The system may include many advanced capabilities, or even easy workarounds or hacks that are possible to accomplish highly time-saving tasks, but most users will ignore these if they are not designed to follow the path of least resistance.

So the question in evaluating an LRS is not necessarily, “What can the system do?” but, “What can the system do in a right-out-of-the-box, plug-and-play, easiest/most-obvious-path use case scenario?” Just because a vendor is able to make a technical case that their system has a particular capability doesn’t mean that it is implemented in a way that is easy for users to see, understand, and use.

5.25 Aligning staff and processes to system capabilities

As with most software, systems that are easier to learn and use generally have fewer capabilities, and vice versa. Sophisticated capabilities will generally require a system that is harder to learn and/or require specialized professional expertise. It is important to determine the skill sets within your pool of LRS administrator staff, so that you know what you are prepared for and/or what you might have to acquire in terms of staffing or training. You can engineer your staff expertise and roles to match the out-of-the-box system, but it is usually not cost-efficient to engineer the system to match staff expertise.

This also applies to task flow; you will almost invariably need to decide whether you want to change your internal processes to match the built-in LRS task flow, or vice versa (i.e., reengineer the LRS to match how your organization does things). This is a complex issue, and there are some strong proponents on the side of choosing an LRS that, out of the box or perhaps with customization, supports your existing processes, but this may be easier said than done. It is likely that you will have to do some of both. Above all, do not underestimate the financial pressure you may find yourself under to tailor your organizational policies and processes to make it easiest to work with the system out of the box. Customization of LRSs, whether open source or commercial products, can be expensive.

The LRS system design and “path of least resistance” workflows can imply changes to your existing processes and infrastructure in the following areas:

- IT infrastructure
- Administrative procedures and policies
- Workplace cultural attitudes and ingrained practices
- Training paradigms

5.26 Planning for operation and governance of your LRS

One important difference between LRSs and LMSs is the fact that LRSs operate behind the scenes, i.e., invisible to end users. They may not in fact know that their interactions within a learning experience are triggering xAPI statements to be sent to the LRS. They will probably never need to see LRS reports or visualizations, except possibly in terms of dashboards that provide feedback on their learning progress, if that is a feature of their learning ecosystem.

Before acquiring your LRS, you should have a preliminary plan for how you will ensure smooth operation of it so that it will be used to its full potential and will address the performance gap that led to your decision to acquire it. You also want to be clear on who will maintain both the system and the content it contains, to avoid confusion and institutional obstacles that could affect the ability of the system to realize its intended mission throughout its lifecycle. Without this preliminary plan, you may face skepticism from management approvers of the acquisition.

This preliminary plan will probably change once your system is fully installed, after you gain some familiarity with it and better understand how to leverage the system features to best express your business needs, processes, and policies. The reverse applies as well as well; you may determine that it is easier to change your processes and policies to match the system's standard features and workflows, as described in *5.25 Aligning staff and processes to system capabilities*.

To ensure a smooth implementation, you need to start this planning during the acquisition phase, not wait until after acquisition. Some aspects of your plan may impact your choice of LRS and vendor, especially if the vendor will act as implementation consultant.

5.27 Bandwidth to the users

For many organizations, bandwidth for traffic in to the LRS server is not much of an issue except perhaps during peak usage times. There are various possible solutions to that problem; a simple one is to stagger learning completion deadlines between courses or groups within a course such that the xAPI statements are not piling up on the server all at once.

However, bandwidth to the users from the LRS could be a problem, especially if they are in remote areas or using BYOD data plans on mobile devices. There could even be a problem within retail outlets, where employees taking training onsite at the outlet are sharing bandwidth with point-of-sale (POS) systems, security systems, customer Wi-fi, etc. With greater and greater use of video (especially high definition) for training, there could be a significant slowdown for all users within the store. This could cause problems for not only trainees who experience latency in their training videos, but for customers trying to complete purchases, etc. in the store.

5.28 Personal data locker

The xAPI spec does not prohibit writing statements to more than one LRS. This raises intriguing possibilities. Perhaps the most obvious use case for this capability is writing statements to an employer LRS and a personal data locker, an LRS account that is "owned" by the employee for the purpose of expanding the scope and relevance of learning experiences beyond the primary purpose for which it was tracked. In other words, multiple LRSs allow the learner to accumulate their own portfolio of learning experiences for later use in meeting requirements for personal or career goals.

5.29 Multiple LRS environments

Within the xAPI spec, activity providers can send statements (concurrently or sequentially) to multiple LRSs. This enables scenarios where LRSs are optimized for particular use cases and needs, and statements are sent to one or the other or both for different purposes and processing. For instance, one LRS may be a statement of record for an organization, and another may be essentially middleware that bounces statements to other business systems. This could in the future result in “boutique” LRSs that are optimized for very particular purposes.

Sending concurrent statements does incur a network bandwidth price that should be taken into consideration; it may be better to send data efficiently in bulk form from one LRS to another on the back end instead of to both from the learning provider front end.

5.30 Open architectures

“Open architecture” infers that the LRS has APIs that allow integration of external applications and systems into the LRS, including, in some cases, swapping an LRS vendor-provided function with an externally produced one. In some cases, the vendor offers hundreds of APIs that the customer can pick and choose from. Open architectures imply a relaxation of proprietary control and constraints on the part of the LRS vendor, allowing potential users to “look under the hood” at their implementation.

To enable open architecture, the vendor usually must share all or parts of its architecture with add-on/system integration developers. This may require some license agreements between entities sharing the architecture information.

Open architecture products tend to have a service-oriented architecture (SOA), and tend to be designed less as closed systems and more as extensible platforms. Because of this, they tend to encourage innovation and experimentation more.

In spite of the potential for competitive disadvantages resulting from publicly exposing the inner workings of their system, some vendors favor them because their customers want to be able to easily customize the system by purchasing additions that the LRS vendor may not feel are important enough to develop themselves.

Open architectures have driven the creation of a substantial marketplace for third-party applications that can be integrated into the core LRS system as modules. These modules can provide all sorts of functions ranging from data visualization engines to providing the capability to transmit data to an ERP system.

Open architectures could significantly decrease risk in cases where changes to your enterprise learning needs and learning technology in general are expected. In these cases, an open architecture can allow you to prolong the useful life of your LRS by incrementally adding needed functionality rather than having to replace it.

As stated in *5.18 Security considerations for LRSs*, it is important to find out what programming language and third party OEM components were used to build the product you are considering acquiring. There are innate security considerations for some programming languages. Also, if you will need to customize the system, your programming staff need to have the skill sets for that programming language and have licensing access to modify any third party components.

5.31 Component-based architecture

In a component-based architecture, a vendor licenses a product for use as an on-demand service—customers pay for only the components they use. It presumes a modular architecture whereby the vendor compartmentalizes the system so that users only access (and pay for) the parts that they need at any given time. This method is attractive to many organizations because it can lower costs (since you only pay for the features you use), in contrast to licensing all applications/modules/functionality 24/7 throughout the life of the installation.

Products with a “service-oriented architecture” (SOA) imply having a component-based architecture, in that services can represent components that can be accessed when needed from the cloud.

Certain aspects of the architecture of such a system must be designed specifically for component-based architecture by the vendor, so that features can be turned on or off, depending on the needs of individual customers. Many current systems offer some degree of component options; qualifying as “component-based” is only a matter of the degree to which the system and the pricing model is optimized for it.

Component-based architectures are usually associated with hosted solutions (see *5.19 Hosting options*). However, a hosted solution may be sold with or without any compartmentalization. For instance, a hosted solution may simply be a one-size-fits-all system based on a flat fee covering a specific number of licenses that cover using all parts of the system; a component-based architecture solution is usually hosted but in addition also involves a modular, compartmentalized approach, as described above.

5.32 Learning Experience Manager

Learning Experience Manager is a new kind of LMS which takes advantage of the xAPI specification’s ability to track learning of all types, including informal and experiential (see **Error! Reference source not found. Error! Reference source not found.**). It remains to be seen whether the name is accepted as an industry standard for this category of product; it is only offered by one company currently (TREK product - <http://www.cognitiveadvisors.com/>). This product provides an LRS endpoint for xAPI communications, allowing tracking of such learning experiences as:

- Coaching conversations
- Searches
- Video watching
- On-the-job experience

It also allows the awarding of badges, learning analytics, ePortfolios, and the creation of individual learning paths. The learning experience manager may represent functionality built onto an LRS, or it may be essentially an LMS with an LRS added; either way, the LRS is a key component.

5.33 Data analytics

Wikipedia cites considerable disagreement among experts as to a definition of learning analytics, but uses as a starting point “...the use of intelligent data, learner-produced data, and analysis models to discover information and social connections for predicting and advising people's learning.” (Wikipedia, 2015). It also differentiates educational data mining from learning analytics, saying that the former is not hypothesis-driven, in contrast to the latter.

Whatever academic definition one chooses, the broad, practical aspects of measurement in a learning ecosystem is something that learning professionals and learning technology vendors are starting to pay more attention to. It is not that measurement was not possible or important before, but now the xAPI spec enables you to capture much finer detail of more parameters in an interoperable way. Through its semantically-based, flexible data model, it facilitates a new level of analysis, by fancy data visualization engines or simple rubberneck checks of data tables, to elegantly answer the perennial questions: Who? What? Why? Where? When? How?

Taking it down a level, here are some examples of the questions that can be answered using a combination of xAPI for data capture and some kind of analytics engine for information output:

- How well am I doing in this learning experience? (individual learners)
- Which learners require or are going to require extra support and attention, and in what specific areas? (instructors)
- What design features of learning experiences are most effective in producing learning in a particular context? (designers)
- What are the most cost-efficient learning interventions? (stakeholders)
- How are particular learning resources actually being used? (content authors and managers)
- What are the best logistical arrangements for marketing and delivering the course? (administrators)

No longer do learning professionals need to be limited to the canned reports produced by LMSs. Traditional LMS reports have served many well for a long time, but we are now in the era of data-driven decision making in the learning space. Data-driven decision-making requires breaking open the black box of data capture and reporting functions within LMSs to provide a much wider range and depth of information than can be provided with predefined reports. LRSs are the on ramp towards separating analytics from course delivery and management functions, which is important given that more and more content is launched and/or experienced outside of the LMS.

With the resurgence of performance support (in many cases, replacing training), there is a greater need for custom data capture and analytics solutions, solutions that are difficult for LMS alone to manage. One simple reason for this difficulty is that performance support is by definition devoid of assessments, which are the primary vehicle for measurement for content in an LMS. Performance support tools require data on whether and how they are being used (called “paradata”), not how much learners have learned from them. Paradata for both performance support and instructional content may include a range of individual or aggregate user interactions such as viewing, downloading, time/place/situational context of use, sharing with others, rating, and using content for derivative products.

Silvers & Torrance (2015), propose the following categories generally related to paradata:

- Sentiment analysis - What do the words people use tell us about their disposition to learn?
- Engagement analysis - What's the activity level with learning content?
- Cohort analysis - Who forms what groups for what reasons?
- Keyword analysis - How do people seek info & what do they find?
- Conversion Rate - How many people respond (i.e., comment)?
- Amplification Rate - How many times is something shared?
- Applause Rate - How many likes/favorites/bookmarks?
- Economic Value - Short/Long Term Revenue/Cost Savings?

LRSs are well suited to answer the above paradata questions, due to the fact that you can use the xAPI to instrument the work environment in addition to the learning environment, and the integration of these analytics can be very powerful in creating feedback loops to fine tune your learning interventions (as well as business processes). Kirkpatrick Level 3 and above evaluations can be more easily institutionalized within your learning ecosystem in this way. The xAPI can not only bridge work and learning data, but it can bridge a learner's physical state over time with learning activities or work performance so that, for instance, heart rate can be correlated with work or learning tasks to determine points of high stress.

Data visualization that allows recognizing complex patterns and trends is an important capability enabled by the xAPI. Because the xAPI allows precise, microscopic statements describing a learner or system's state at a specific point in time, trends can be seen easily with graphs, diagrams, etc. You do not need to create expensive custom visualization engines to do this. Open source solutions are available such as ADL's xAPI Dashboard (<https://github.com/adlnet/xAPI-Dashboard>) and Apereo Open Dashboard (<https://github.com/Apereo-Learning-Analytics-Initiative/OpenDashboard>).

The obvious, traditional approach to analytics is to plan your analytics solution (using xAPI) to answer specific questions first, then capture data. This works where you have specific measurement needs that are already clearly defined, usually resembling the typical reports provided by LMSs. For those inclined towards thinking of data analytics more as research and data mining (ie, without an initial hypothesis, as mentioned above), the xAPI provides a durable, interoperable basis for analytics engines to create visualizations that can reveal unexpected patterns. One can think of this approach as "measure first, ask questions later" - in other words, capture lots of different kinds of data just because you can, and then explore to see what emerges from it from analysis. The steps for this kind of research-oriented approach could be:

1. (optional) Formulate baseline research questions. You need some idea of these, even if you are using this exploratory approach, as a basis for Step 2 below and data analysis/visualization methods later.
2. Decide what interaction nodes and learner behaviors in the learning experience make sense to instrument with xAPI.
3. Decide what granularity you need and the right syntax and verbs for your xAPI statements. This is essentially becomes your hypothesis, if you are using one.

4. Deploy xAPI-instrumented learning experience and collect data.
5. Validate data received against research questions
AND/OR
Look for patterns
6. Refine xAPI granularity, verbs, LRS queries, etc.

For those inclined towards data modeling and “what if” scenarios, not only can historical data be collected and subjected to various analyses after the fact, but specific hypothetical data (ie, xAPI statements) can be substituted for real historical xAPI statements. The xAPI allows you to insert these hypothetical statements in a surgically precise way and then play out the scenario in your data analytics engine, to see what results could emerge that are different from the real results.

6. Criteria for assessing quality and suitability of LRSs

The following is a list of characteristics, features, and functions that a robust LRS should include. The applicability of items in this list to your situation will probably vary widely; some items may be mission-critical for your organization and some may not be pertinent at all. You need to carefully weigh the importance of each in evaluating LRSs. If you rate your list of LRS candidates simply by all items in the list without weighting each item for its importance to you, it could skew the results, which could lead to a poor final choice for your system.

There is also the issue of the degree of support that the LRS provides for a certain feature. Very few of the features listed below are either 100% present or 100% absent in a given LRS.

The following is a list of features that could be included in a robust LRS. Some of the items in the list are commonly implemented and some are developmental. The list is intended to represent items you could look for in an LRS (though possibly not easily find).

- System access and security
 - Conforms to secure application infrastructure standards such as ISO/IEC 27001
 - Offers the ability to send test statements to verify connection
 - Allows encryption of sensitive data (i.e., passwords) and session activity (i.e., LRS-related network traffic)
 - (for government organizations) Conforms to applicable security regulations such as 21 CFR Part 11, EU GMP Annex 11
 - Affords a high level of password security features, for instance:
 - Allows the administrator to require users to use strong passwords
 - Limits the use of old passwords
 - Defines parameters for strong passwords for users
 - Requires users to change the password on first login
 - Locks users out after a certain number of failed login attempts
 - Requires users to change passwords regularly (using notifications)

- Sets limits on periods of inactivity
- Only users can change their password
- Encrypts stored passwords
- Is able to handle digital signatures. In a government installation, this could require compliance with federal regulations like 21 CFR Part 11.
- Provides a single sign-on, so that users who have logged in to the enterprise intranet (through a portal, etc.) can get into the LRS without additional login
- Allows login to the LRS to transfer to other enterprise systems (especially HR)
- Requires user logon only once per LRS session
- Requires each user to be uniquely identifiable (e.g., user name or user ID)
- Runs all user requests through a common security checkpoint in the system architecture
- Was developed by a single company (the vendor), to avoid risks associated with exposure of code to external organizations
- Provides audit trails for changes to data in the system such that the organization can quickly determine the source of unauthorized activity that could be the source of security breaches. These changes could include everything from uploading learning objects to running reports.
- Supports industry-standard authentication using such standards as:
 - LDAP
 - CAS
 - Shibboleth
 - Kerberos
 - SSO SAML
 - Social logins (Facebook, Gmail, etc.)
 - (For high-security government installations) Allows Common Access Card (CAC) access
- Incorporates appropriate security certifications and standards, and features (see 5.18 Security considerations for LRSs and 5.20 Special requirements for U.S. DoD). Other security standards you may need include SSL, PKI, and FIPS – 140-1.
- Allows configuration for the management of PII in accordance with enterprise and government policy (such as FERPA)
- (for DoD) Contains multiple security access levels with ready access to unclassified learning material and more stringent security requirements for FOUO and classified information
- (for hosted solutions) The provider:

- Has Intrusion Detection/Prevention services
- Monitors individual LRS instances for suspicious activity, in real time
- Regularly audits the security of its servers
- System performance
 - Has a quality assurance process whereby changes to the xAPI spec or the LRS product are regression tested with an internal test suite to ensure strict compliance with the spec
 - Performs with minimal latency under a variety of use case scenarios and load conditions
 - Handles large numbers of concurrent users
 - Can handle surges in volume of statements sent to it
 - Handles user load efficiently, provisioning and scaling resources to smoothly accommodate fluctuations (especially spikes) in numbers of concurrent users
 - Enables Web hooks. i.e., push-based service requests. For instance, if you are making an xAPI-enabled reporting app, and you want to be alerted every time a statement comes in that is relevant to what the app is reporting on, you can configure the LRS to send any statements that come in that fit the criteria for relevant statements. The LRS can get statements from different places, but every time it gets one that is relevant, it is pushed to the reporting app.
 - Enables Web sockets (a different protocol than HTTP). The analogy here is that the standard way to communicate with an LRS is like mailing a letter (the statement). A Web socket is like a phone call. A Web socket allows much higher throughput. Not all systems support it, because some programming languages don't support it.
 - Works equally well on all standard Internet browsers, including a reasonable span of legacy versions of those browsers (backward compatibility with 2 year-old versions is often used as a rule of thumb)
 - Has reasonable system requirements that are attainable within your organization
 - Uses normalized architectures for hardware and software implementations
 - Can be load balanced across multiple servers
 - Can be clustered
 - Has robust mechanisms for coping with machine failure such that no loss of data occurs
- Cost
 - Offers a free trial
 - Costs less for the base application license compared to the cost of other similar systems with similar capabilities and feature sets. This includes all TCO (total cost of ownership) costs.

- Has a licensing agreement that is flexible and easily scalable to reflect changing numbers of learners and administrators. This is especially important if you project substantial growth in your organization, or have “extended enterprise” users
- Allows you to meter usage of the system by individual business units, so that you can spread the cost fairly
- Costs less for recurring and ongoing support compared to the cost of other similar systems
- Is projected to cost less for required customizations compared to the cost of customizations for other similar systems
- Costs less for add-ons such as APIs to external applications compared to the cost for this in other LRSs
- Offers hosted (also termed SaaS or cloud) and/or component-based architecture solutions to take advantage of these potentially cost-saving options (see *5.19 Hosting options* and *5.31 Component-based architecture* for details)
- Costs minimally extra for separate test and staging instances of the product (see *5.21 Test and staging environments*)
- Uses or can use open source components that can significantly reduce costs
- Has a vendor who is open to cost sharing arrangements. If you are planning to make extensive customizations, discuss with the vendor possible partnering on the development and/or cost of such changes so that the cost or development can be shared with the vendor and/or other customers, if other customers who have purchased the vendor’s product will receive the new functionality.

It is standard practice for vendors to use customer requests for customization as an economic stimulus for their development of new system features, such that the cost of developing these features (that are included in system upgrades that everyone gets) is effectively funded by these customers.

- System integration
 - Includes data migration tools for moving data permanently from an LRS to and from another LRS or an LMS
 - Supports LRS to LRS real time communication
 - Enables Comma Separated Values (CSV) or Excel data export
 - Interfaces with systems that you might have in your enterprise such as:
 - Learning systems
 - LMS
 - LCMS
 - VLE
 - CrMS

- Learning content repositories
- Third party course content
- Learner registration system
- Any system from which you will need to import legacy learner tracking data
- HR systems
 - HR database
 - Performance management systems
 - Talent management systems
- Enterprise resource planning (ERP) systems
- Business intelligence systems
- Intranets
- Marketing, sales, CRM, and financial platforms
- Content repositories
- Document management systems
- Collaboration tools
- IT administrative systems
- Authentication systems
- Authorization systems
- Data validation systems
- Email directories
- Imports and exports to external systems in real-time and batch mode
- Enables add-ons and integration using an open architecture (see 5.29 *Multiple LRS environments*)
- Import and export of learner and course tracking data using standardized data interchange formats (e.g., XML, JSON, CSV) without writing high-LOE integration applications
- Links to employee records in an external system
- Analytics, visualizations, and reports
 - Includes the ability to add custom fields to track additional learner information
 - Includes RESTful APIs that allow custom views of LRS analytics
 - Provides the ability to print a variety of tracking-related items, including test scores
 - Offers xAPI tracking and display of “bookmarklets”

- Allows drill down to actual statements (with filters/ search by Activity ID, Verb ID, Agent Value, Agent Property).
- Statement viewer function allows filtering by organization hierarchy or custom defined group.
- Offers a wide variety and number of predefined reports and visualizations
- Offers permission levels with different kinds of access to reports.
- Offers flexible, robust abilities to create custom reports, both internally and by using external tools (including those supplied by other vendors such as Crystal Reports®)
- Prints reports easily, with appropriate options
- Provides the ability to embed reports (through a Report API) into other tools and systems, using standards like LTI
- Allows sharing of report or visualization (snapshots or real time) by interested parties via sharing of a direct URL
- Provides capabilities to:
 - browse xAPI statement data
 - use canned reports for commonly required data such as test scores
 - use configurable, customizable reports
 - measure business impact (through integration with external systems possibly)
- Provides direct access to tables used within the LRS for developing queries and reports. This should be documented in table and data structure specifications provided with the product. This is usually a requirement for government installations.
- Provides reporting on certification status of groups and individuals, including upcoming renewals, missed renewal deadlines, etc.
- Provides ways to incorporate data from external systems to produce reports and analytics that show “big picture” measures of employee learning progress activity across all knowledge transfer mechanisms
- Provides dashboards for learners to give them feedback on their progress
- Provides personal “bookmarklets” (location indicators learners can place in the content which are recorded via xAPI)
- Includes ROI analytics-oriented analytics
- Includes ability to tag users with their position in the organizational hierarchy, and filter by this attribute in reports and analytics. Tags can be set within xAPI statements or set in the user profile residing within the system.
- Provides standard learning content-centric reports that include at a minimum:
 - Average time spent by users in the content
 - First and last access of content
 - Number of downloads or user access

- Number of attempts to pass
- Successful attempts to pass
- Type of content (media file, elearning module, etc.)
- Expiration or deadline for users to take content
- Provides user-centric reports that include at a minimum:
 - Date items (includes content objects, tracks, etc.) assigned (by admin or instructor)
 - Date of start by user
 - Date of completion by user
 - User score
 - Result
- Includes analytics engines that include graph charting and advanced visualization options
- Supports creation and sharing of Mozilla Open Badges
- Offers features addressing the range of analytics maturity models (NetDimensions, 2015)
 - **Passive reporting** – canned reporting and dashboards
 - **Proactive reporting** – KPI reporting and dashboards
 - **Siloed analytics** – domain-specific relationship analysis
 - **Integrated analytics** – across HR/talent domain and business aligned/ connected
 - **Predictive analytics** – dynamic future scenario modeling
 - **Machine intelligence automation** – dynamic automated data-drive decision making and machine action
- Ease of use for administrators
 - Is easy to learn and use, with the ability for users to choose from tiers of features according to the knowledge and expertise of the user. This allows users to start using the program quickly and gradually progress to more complex authoring tiers/feature sets as their skills mature. In other words, users only see features that are relevant to their level of skill and the kind of operations they are capable of performing. Ease of use for administrators is important since it can reduce the skill set requirements and thus the cost of administrators.
 - Provides user interface customization (not on the level of tiers of features, as above, but on an individual feature basis), so that both learners and administrators can optimize for their particular needs
 - Can easily create and restore archives of system (e.g., administrative actions) and user (e.g., xAPI statements), in a proprietary or open format

- Is easy to install and reconfigure
- Manages the administration process efficiently with built-in workflows (for approvals, for instance)
- Administrative interfaces are clear, simple, and optimized for usability.
- Includes options for remote administration from outside the enterprise intranet (through the Internet) and possibly via a mobile device
- Provides features that allow administrators to view role structures in a graphical representation (diagrams, outlines, etc.)
- Provides clear, specific error messages that aid in troubleshooting. A generic message that is the same for all errors is not acceptable. You also want to avoid cryptic, technical messages that can only be interpreted by the LRS's software developers. Messages should be understandable not just to technically inclined LRS administrators, but also to xAPI activity providers who want to do testing of content. Also, it is ideal for error messages to vary depending on whether you are in the test vs. the production system.
- Has a feature to store favorite locations within the system
- Allows saving of a workspace
- Scalability
 - Has a scalable architecture that allows the system to expand as the number of users increases. The following factors should be taken into account in your planning:
 - Number of concurrent users (current and in the foreseeable future)
 - xAPI statement volume restrictions
 - Supports large media attachments (e.g., videos) that can be attached to xAPI statements
 - Has a scalable architecture, enabling evolution of the client installation without forcing them to go through frequent major version upgrades
 - Allows configuration of a data distribution network (interconnect distribution peers through a common distribution server)
- Vendor characteristics
 - Has consulting experience and service arrangements. This is especially critical for LRSs, since the data-driven learning paradigm and xAPI supporting it is so new. You may need extensive help in designing ecosystem-wide solutions if you are new to this space. Your consulting needs will probably not just be limited to getting help configuring or using your LRS.
 - Has a good reputation among acquisition and system owner communities. Ask the vendor who their other clients are, what they use the system for, and see if you can talk to these clients about their experience using the system. Look for negative comments posted on the Internet by members of these communities.

- Is willing to openly discuss with you changes in the learning landscape, and their roadmap for adapting to these changes.
- Has been in the LRS market for at least a year. Avoid the first release of a new system.
- Has not created the product merely as an add-on to an ERP or some other system, in order to be able to sell it to customers desperate to add an LRS to their existing system. Although the cost will probably be lower than purchasing a separate LRS, and the system will obviously be well-integrated with the ERP, it can mean that the LRS receives short shrift in design and usability.
- Has a clear technology roadmap with a reasonable time frame for new versions and additions of new features
- Listens to your concerns during interactions with them, especially during demo sessions of their product. How they are in these situations probably reflects how responsive and attentive they will be to your concerns as a customer.
- Is financially sound and not in danger of going bankrupt. You may want to consider acquiring Dun and Bradstreet reports for your final vendor candidates, to establish the financial health, stability, and long term business strategy of them.
- Is of a stable size, as measured by number of employees, annual revenue, capitalization, etc.
- Has a robust ongoing budget for R&D
- Has a large number of successful clients. Who the clients are and their industry stature can be important, especially in terms of their similarity to your mission or infrastructure. If you can, find out the number of total users served by the LRS product within this client base.
- Is not about to be acquired or merged with another vendor. Obsolescence and durability is an important consideration in the fast-changing landscape of LRSs and enterprise systems in general. You don't want a vendor that gets bought out by another company, and your LRS, with all of your expensive customizations, no longer functions because it has been reengineered to conform to the acquiring vendor's architecture, or worse, has been withdrawn from the marketplace because it is redundant with a product that the acquiring vendor already has in place.
- Has worked with many content developers using a variety of different kinds of content. Ask for references at organizations that have deployed content similar to yours.
- Is familiar with your business model, market, and content types.
- Is International Standards Organization (ISO) and/or Capability Maturity Model Integration (CMMI) certified to ensure high-quality software development output
- User training, technical support, and documentation
 - Has robust support for training of all categories of users: learners, instructors, system administrators, content managers, etc.

- Has robust support documentation in a wide variety of forms including tutorials, help, examples, references, and user manuals
- Has a variety of Help Desk support options for administrators and learners (telephone, chat, email, etc.). These need to be in synch with the way your organization normally requests help.
- Has a Help Desk system that is structured and process driven via trouble call tracking and reporting
- Has Help Desk support that coordinates problem resolution with the appropriate parties: vendors, SME's, etc. for problem resolution
- Has knowledgeable, experienced support personnel
- Is available as close to 24/7 and world-wide as possible
- Offers extensive training options: eLearning, video tutorials, ILT sessions, webinars, etc.
- Has onsite training options. If training is at vendor site, the location(s) are a reasonable distance.
- Includes an orientation tutorial for new users
- Has a low average turn-around time for Help Desk support
- Has a feedback function for suggestions on improving the LRS
- Provides technical consulting services options for customizations, implementation, configuration, architecture design, needs analysis, change management services, etc.

7. For more information about LRSs and xAPI

7.1 ADL tools and resources

- ADL xAPI Resources home
<http://adlnet.github.io/#xapi>
- Overview of xAPI
<http://www.adlnet.gov/tla/experience-api/background-and-history.html>
- ADL Learning Record Store (LRS)
https://github.com/adlnet/ADL_LRS
- Installation tutorial instructions
<http://adlnet.github.io/xapi-cohorts/season-3/tutorials/adl-lrs/quickinstall/index.html>
- Demo of an installation [video]
https://www.youtube.com/watch?v=WBSIJ-Fc_Fw
- The actual xAPI specification
<https://github.com/adlnet/xAPI-Spec>

- xAPI Verbs
<https://github.com/adlnet/xAPIVerbs>
- The xAPI Wrapper
<https://github.com/adlnet/xAPIWrapper>
- xAPI Dashboard
<https://github.com/adlnet/xAPI-Dashboard>
 - Overview [video]:
<https://www.youtube.com/watch?v=f-iZQ-ykXN8>
- xAPI Lab
<https://github.com/adlnet/xapi-lab>
 - [video]:
<https://www.youtube.com/watch?v=TNCHYMpnpJ0>
- xAPI Statement Viewer
<https://github.com/adlnet/xapi-statement-viewer>
- xAPI jQuery Mobile
<https://github.com/adlnet/jxapi>
- SCORM-to-xAPI Wrapper
<https://github.com/adlnet/SCORM-to-xAPI-Wrapper>
 - Examples of SCORM-to-xAPI wrapper use:
<https://github.com/adlnet/SCORM-to-TLA-Roadmap/tree/master/examples/RosesCourseConversion>
- xAPI Canteen
<https://github.com/adlnet/xapi-canteen>
- xAPI Remarks
<https://github.com/adlnet/xapi-remarks>
 - Overview [video]
<https://www.youtube.com/watch?v=LmWv-5muuM0>
- xAPI YouTube
<https://github.com/adlnet/xapi-youtube>
- xAPI Java Library
<https://github.com/adlnet/jxapi>
- xAPI adopters lists
<http://www.adlnet.gov/tla/experience-api/adopters.html#adopters>
- xAPI Client Examples
 - ADL
https://github.com/adlnet/experienceapi_client_examples
- xAPI SCORM Profile
<https://github.com/adlnet/xAPI-SCORM-Profile/blob/master/xapi-scorm-profile.md>

- [video]
<https://youtu.be/nxkdzYMsegQ>
- xAPI Design Cohort (designed to help people understand how to design and develop for use of the Experience API):
 - Overview
<http://www.adlnet.gov/from-adl-team-member-craig-wiggins-xapi-design-cohort-season-3-kickoff/>
- xAPI Design Google Group
<https://groups.google.com/a/adlnet.gov/forum/#!/forum/xapi-design>
- Resources from current and past participant groups [under construction]:
<http://adlnet.github.io/xapi-cohorts/>
- xAPI Specification Google Group:
<https://groups.google.com/a/adlnet.gov/forum/#!/forum/xapi-spec>
- Writing about xAPI by ADL members
 - The Experience API—Liberating Learning Design:
<http://www.elearningguild.com/research/archives/index.cfm?id=177&action=viewonly>
- Five Things a Web Developer Needs to Know About the xAPI
<http://www.learningsolutionsmag.com/articles/1526/five-things-a-web-designer-needs-to-know-about-the-xapi>
- Ten Steps to Plan and Communicate Your xAPI Design to a Web Developer
<http://www.learningsolutionsmag.com/articles/1523/ten-steps-to-plan--communicate-your-xapi-design-to-a-web-developer>
- Establish an xAPI Infrastructure: Guide to Gain Sponsorship & IT Support
<http://www.learningsolutionsmag.com/articles/1582/establish-an-xapi-infrastructure-guide-to-gain-sponsorship--it-support>
- xAPI and Analytics: Measuring Your Way to Success
<http://www.learningsolutionsmag.com/articles/1722/xapi-and-analytics-measuring-your-way-to-success>
- Are You an ISD? A Business Process Engineer? Or Both?
<http://www.learningsolutionsmag.com/articles/1745/are-you-an-isd-a-business-process-engineer-or-both>
- xAPI Communities of Practice
 - Google Group
<https://groups.google.com/a/adlnet.gov/forum/#!/forum/xapi-profile-cop>
 - Directory
<http://www.adlnet.gov/tla/experience-api/xapi-cop-directory>
 - Videos + xAPI
<http://goo.gl/JTrqmn>

- Augmented Reality + xAPI
<http://www.juxtopia.org/programs/xapi-ar-program/>
- Open Badges + xAPI
<https://github.com/ht2/BadgesCoP>
- xAPI in the context of Chinese language and culture (page in Chinese, naturally):
<https://www.facebook.com/groups/648340368618407/>
- Elearning courses + xAPI
<http://adlnet.github.io/xAPI-SCORM-Profile/index.html>
- ebooks + xAPI:
<https://ieee-sa.centraldesktop.com/adb/>
- Medical education + xAPI
<http://www.medbiq.org/medbiq/display/XIG/XAPI+Interest+Group+Home>
- Simulations + xAPI
<https://plus.google.com/u/0/communities/103100550395134532597>
- Social Collaboration + xAPI:
<https://plus.google.com/u/0/communities/112155587504087853094>

7.2 Non-ADL resources

- Overview
 - KMI Learning[video]
<https://youtu.be/y42MSS1DJqc>
 - Float Learning
<http://floatlearning.com/xapicartoon/>
 - Rustici Software
<http://tincanapi.com/the-layers-of-tin-can/>
- How xAPI is being used
 - Slide presentations
 - <http://www.slideshare.net/RusticiSoftware/the-impacts-of-the-tin-can-api-how-8-companies-are-using-the-tin-can-api-xapi>
 - <http://www.slideshare.net/RusticiSoftware/nine-applications-of-the-tin-can-api-xapi>
 - Q&A from live sessions
 - <http://tincanapi.com/2015/02/10/webinar-qa-impacts-tin-can-api-8-companies-using-tin-can-api-xapi>
 - <http://tincanapi.com/2015/04/07/webinar-qa-nine-practical-uses-tin-can-api-xapi/>
 - Videos

- <https://youtu.be/jSQwo1C4feA>
- <https://youtu.be/8LFhDdqQ13A>
- Adopters List
<http://tincanapi.com/adopters/>
- Events
 - DevLearn 2014 (Las Vegas)
<http://www.elearningguild.com/DevLearn/content/3608/devlearn-2014-conference--expo--spotlight-on-xapi/>
 - xAPI Bootcamp 2014 (Orlando FL): held at ADL Orlando offices
<http://www.adlnet.gov/resources-available-from-june-2014-xapi-workshop-bootcamp-in-orlando/index.html>
 - xAPI PlugFest 2014 (@ I/ITSEC, Orlando FL)
<http://www.adlnet.gov/iitsec-xapi-plugfest-2014/index.html>
 - xAPI Camp UCF 2015 (Orlando FL)
<http://connectionsforum.com/xapi-camp-march-2015/> (also, see my Storify, assembled from afar)
 - TryxAPI (@ ATD ICE, May 2015, Orlando FL)
<https://roundtown.com/event/7006998/TRYxAPI-Launch-Party-Orlando-FL>
 - xAPI Tin Can Meetup (@ ATD ICE, May 2015, Orlando FL)
<http://www.eventbrite.com/e/xapi-tin-can-meetup-tickets-15814460477?aff=eventful/r/eventful>
 - xAPI Bootcamp 2015 (Alexandria VA, July 14-15, 2015):
<https://github.com/adlnet/xapi-bootcamp-2015/wiki>
Contact craig.wiggins.ctr@adlnet.gov for more information.
 - xAPI Camp Amazon 2015 (Seattle WA, July 21, 2015):
<http://connectionsforum.com/xapi-camp-seattle>
 - xAPI Camp DevLearn 2015 (@ DevLearn, Las Vegas NV, September 29, 2015):
<http://connectionsforum.com/xapi-camp-devlearn-2015>
- Tin Can (xAPI)
 - Registry
<https://registry.tincanapi.com/#content/about>
 - Recipes
<http://tincanapi.com/recipes/>
- CMI5
 - What is cmi5?
https://www.youtube.com/playlist?list=PLlv_yyODMQs5t2aom5cYp1Z5Dv6BxMVkg

- The specification:
 - https://github.com/AICC/CMI-5_Spec_Current
- Experience API, cmi5, and the Future of SCORM
 - <http://www.learningsolutionsmag.com/articles/1697/experience-api-cmi5-and-future-scorm>
- LRS-to-LRS communication
 - How to share statements (blog post)
 - <http://tincanapi.com/2015/04/30/share-statements-how/>
 - Sharing xAPI statements between Learning Record Stores (video)
 - https://youtu.be/I563fetvX_8
 - Sharing Between LRSs: A Collaborative Experiment in practical interoperability (whitepaper)
 - <http://tincanapi.com/wp-content/uploads/2015/03/whitepaper.pdf>
- Multiple LRSs
 - <http://blog.saltbox.com/blog/2012/06/20/tin-can-one-possible-future/>
- LMSs vs LRSs
 - <http://blog.saltbox.com/blog/2012/10/30/lms-or-lrs/>
- Why/how LMSs should add LRSs and xAPI support:
 - <http://blog.saltbox.com/blog/2013/09/20/lms-innovation/>
- Integrating LRS with LMSs (or "Learning Platform Providers")
 - <http://blog.saltbox.com/blog/2015/03/02/introducing-wax-lrs-for-learning-platform-providers-a-white-labeled-lrs-and-reporting-engine/>
- How some LMSs are acting like more of an activity provider and giving customers the choice of using their own LRS
 - <http://blog.saltbox.com/blog/2015/06/25/learnupon-lms-supports-xapi-tracking-to-wax-lrs/>
- Why you want reporting capabilities closely attached to the LRS (or in the LRS)
 - <http://blog.saltbox.com/blog/2015/09/23/why-reporting-in-the-lrs/>
- Collections of resources
 - A collection of resources (from Shelly Blake-Plock of Yet Analytics)
 - <https://flipboard.com/@blakeplock/experience-n27f7ge4z>
 - A collection of xAPI resources (from Saltbox)
 - <http://www.saltbox.com/experience-api-resources.html>
 - An xAPI + IoT demo (from Team xAPI Gnome, a participant group in the current xAPI Design Cohort)
 - <https://www.youtube.com/watch?v=7vATddfzXU>
 - SCORM-to-TLA Roadmap (explaining how one might move beyond SCORM):
 - <http://adlnet.github.io/SCORM-to-TLA-Roadmap/>
- Connections Forum
 - <http://connectionsforum.com/vision/>

8. References cited in this paper

- ADL (2013). Experience API specification. Retrieved 7/7/15 from <https://github.com/adlnet/xAPI-Spec/blob/1.0.1/xAPI.md#def-learning-record-store>
- Berking, P., Foreman, S., Haag, J., Wiggins, C. (2015). The Experience API—Liberating Learning Design. Elearning Guild Hot Topics research report. Retrieved November 4, 2015 from <http://www.elearningguild.com/research/archives/index.cfm?id=177&action=viewonly>
- NetDimensions (2015). LMS: Evolution or Extinction – 8 Trends that Change Everything. White paper retrieved 6/18/15 from <http://www.netdimensions.com/downloads-2015/infographics/lms-evolution-or-extinction.pdf>
- Rustici Software (2015a). The Impacts of the Tin Can API: How 8 Companies are Using Tin Can (xAPI). Slide deck retrieved 8/17/15 from <http://www.slideshare.net/RusticiSoftware/the-impacts-of-the-tin-can-api-how-8-companies-are-using-the-tin-can-api-xapi>
- Rustici Software (2015b). Layers of the Tin Can (xAPI) Onion (2015). Web article retrieved 8/17/15 from <http://tincanapi.com/the-layers-of-tin-can/>

Appendix

8.1 Sample System Requirements Matrix

The following is a sample of a matrix that can be used in step 7 presented in *3 Process for choosing an LRS*. The step is described as:

Develop and populate a system requirements matrix that allows assessing the systems identified in step 6 against your requirements developed in step 3. To use the matrix:

Enter items you have determined to be your high-level requirements for the system as row labels in the “High-level requirements” column.

Enter the product names at the top of each column, replacing “LRS product 1”, “LRS product 2”, etc..

Research and complete the cells with information indicating whether each product meets that requirement (may be “yes” or “no”, a more lengthy description of how it meets or doesn’t meet the requirement, or a number that roughly quantifies the degree to which that requirement is supported in the product).

8.2 Sample System Features Rating Matrix

The following is a sample of a matrix that can be used in step 10 presented in *3 Process for choosing an LRS*. The step is described as:

Develop a system features rating matrix...that compares the systems identified in step 8 using the features list developed in step 9. Complete as much of this matrix as possible from the systems' documentation; if you need more information, ask their sales representatives for it (though beware of overblown claims—verify lofty ones independently if possible). Assign a numerical rating for each cell in the matrix, indicating degree of implementation of that feature; “0” would indicate that a particular LRS does not have that feature, and “10” indicates that it has a very robust implementation of the feature. The matrix should weight each feature according to its importance to you, enabling a rollup score for each system.

To use the matrix:

1. Replace the top row (LRS product 1, LRS product 2, etc.) with the names of the systems you have identified for consideration.
2. Replace the row names (Feature 1, Feature 2, etc.) with the names of features you have identified as requirements.
3. For each Weighting factor cell in the column to the right of the Feature name, replace the text with a number between 1-3 to weight the relative importance of that feature to your organization (the higher the number, the more important).
4. Research the feature information for each system and complete the cells with the number indicating the degree to which each system has that feature. We suggest 0-2, 0 being “does not have that feature” and 2 being “has implemented this feature to the fullest extent possible”. You may want to use a rubric developed by Brandon-Hall (Brandon-Hall Group, 2010) that rates the feature in terms of how “out of the box” it is. Assigning numbers to their rubric would yield the following rating scale:
 - 5=Automatic (built-in, out of the box feature)
 - 4=Semi-automatic (mostly built-in, but requires some programming or customization to activate)
 - 3=Semi-custom (partially available. The system can be adapted to implement this feature through moderate customization)
 - 2=Custom (not available but can be added, possibly at high cost, with programming)
 - 1=Not available (would be impossible or cost-prohibitive to customize the system to add the feature due to incompatibilities with system architecture, etc.)

If a feature is not available, you may also want to note in this matrix whether a feature is available from another vendor as an add-on, so as not to totally rule out/penalize the vendor for lack of that feature. This can be incorporated into the rating scale such that a rating of “3” means that a feature is available as a third party add-on.

The rollup score row at the bottom will provide the total weighted score for each system (right-click on it and select **Update Field** after you make any changes to the weighting values or ratings).

If you add columns or rows, copy and paste the Rollup score formula and adjust the row and column references in the formula accordingly. Right-click the pasted Rollup score and select **Toggle Field Codes** to see and edit the formula.

| LRS Features Rating Matrix | | | | | | |
|----------------------------|------------------|---------------|---------------|---------------|---------------|---------------|
| Feature name | Weighting factor | LRS product 1 | LRS product 2 | LRS product 3 | LRS product 4 | LRS product 5 |
| Feature 1 | | | | | | |
| Feature 2 | | | | | | |
| Feature 3 | | | | | | |
| Feature 4 | | | | | | |
| Feature 5 | | | | | | |
| Feature 6 | | | | | | |
| Feature 7 | | | | | | |
| Feature 8 | | | | | | |
| Feature 9 | | | | | | |
| Feature 10 | | | | | | |
| | Rollup score | 0 | 0 | 0 | 0 | 0 |