Learning Tools Interoperability® (LTI®): A Best Practice Guide

by

Stephen P Vickers
Simon Booth

for

ceLTIc:developers Project

celtic-project.org

August 2014

e: stephen@spvsoftwareproducts.com

Jisc

Contents

Case studies

1 Introduction

The Learning Tools Interoperability (LTI) specification was first released by the IMS Global Learning Consortium as Basic LTI in May 2010 [IMS-A] (and is now referred to as LTI 1.0). Since that time it has become well adopted as a simple but effective mechanism for integrating third party content and products with virtual learning environments (VLEs). For example, it is now part of the core product for all major VLEs, including Moodle, Blackboard Learn 9, Desire2Learn and Canvas from Instructure.

The purpose of this document is to provide you with a detailed exploration of how LTI works and the issues which arise so as to give you guidance on good practice, if not, best practice. The document contains sections relating to all the main parties involved in delivering learning experiences to students:

- developers;
- VLE administrators;
- teachers;
- service providers.

Please provide feedback based on your own experiences to help improve the quality of the recommendations included here.

1.1 Terminology

We start with a bit of terminology to help make the descriptions which follow more precise. The LTI specification uses the following terms:

- *tool consumer* - typically this refers to the VLE; it is the system which users are logged into and from which they will be redirected to the external applications being integrated with it;
- *tool provider* - this is the web-based learning application or content delivery system which is being integrated with the tool consumer;
- *consumer key* - this string value is generated by the tool provider to allow them to uniquely identify the source of requests being received;
- *shared secret* - the communications between the tool provider and the tool consumer are secured using a signature generated using the OAuth 1.0 protocol [OAuth-A] with the shared secret (which should be known only to the tool provider and the tool consumer);
- *context* - tool consumers are typically organised into courses, with users being enrolled into each course for which they are permitted to access; thus, in this case, the course is the context from which users launch into tool providers;
- *resource link* - this is the actual link provided within a context which users follow to access the tool provider; there may be multiple resource links to each tool provider within the same context and across the whole tool consumer but each resource link is uniquely identified.

1.2 LTI Releases

This document provides guidance for both the tool consumer and the tool provider components of an LTI 1 connection. At the time of writing, the following versions of LTI 1 have been released:

Version	Release date
1	May 2010
1.1	March 2012
1.1.1	June 2012
1.2	Public draft
2.0	January 2014

LTI version 2 provides a more extensible framework for implementing learning application/content integrations, but this document will focus essentially on LTI 1 because this is what is currently in use and widely available. At the time of writing LTI 1.2 is in public draft and extends LTI 1 with some of the features of LTI 2.0 (e.g. a tool consumer profile enabling discoverable services) thereby offering a stepping stone for systems looking to move from LTI 1 to LTI 2. [IMS-A]

2 Issues for Developers

This section discusses issues relevant to those seeking to write an LTI tool provider application.

2.1 Anatomy of a launch request

The essence of LTI is the launch request. This is the mechanism used by a tool consumer to redirect a user to a tool provider. The fundamentals of this process are:

- redirection is via the user's browser using an HTTP POST request;
- data is passed using POST data parameters with prescribed names;
- parameters may include data about the context and resource link from which the launch request originates, the user making the request, and the role of the user within the context;
- the connection is secured by a timestamp, a nonce value and an OAuth signature.

A tool consumer will typically implement a launch by returning an HTML page to the user's browser consisting of a form containing all the launch parameters as hidden input elements. The form would be submitted automatically by a JavaScript function run when the page is loaded or, if JavaScript is not enabled, by the user clicking a submit button. In most cases, however, JavaScript is a requirement for the tool consumer and so the launch process should be invisible and seamless to the user.

2.2 Launch parameters

The names of supported parameters can be found in the LTI specification. [IMS-B, Sections 3, 4.2 and 6] The only required parameters are:

- lti_message_type
- lti_version
- resource_link_id

plus the following OAuth parameters:

- oauth_consumer_key
- oauth_signature_method
- oauth_timestamp

- oauth_nonce
- oauth_version
- oauth_signature
- oauth_callback

So, as you can see, a launch request can be extremely minimal and need not include any data about the context, user or role. When a tool consumer or tool provider has been certified by IMS, some of the parameters recommended by the specification will also be present. These additional demands for certification are designed to enhance interoperability, whilst still allowing flexibility in the specification.

2.3 Security

Launch requests are secured by the tool consumer to allow the tool provider to verify their authenticity. The launch parameters include:

- *oauth_consumer_key* – the unique identifier for the tool consumer;
- *oauth_timestamp* – the current Unix epoch time from the tool consumer server;
- *oauth_nonce* – a unique value for the request;
- *oauth_signature* – a hash of the data included in the message generated using a shared secret.

The consumer key is generated by the tool provider who should ensure that all the keys issued are unique and are used only for a single tool consumer system. For example, separate keys should be issued for development, test and production environments. The unique keys may be generated in any manner; for example, based on the client's domain name, or a random GUID.

The timestamp should be checked to ensure it is current (within a small margin of error from the system time on the tool provider server). Any request received which re-uses a nonce value (from the same tool consumer) within the permitted timestamp range should be rejected. The signature should be re-generated by the tool provider using the shared secret and compared with the value received. The shared secret should be a random string of a reasonable length to prevent it from being easily guessed or recreated. It should be shared with the tool consumer in a secure manner so as to maintain its privacy between the two parties.

These mechanisms ensure that the data in requests has not been changed in transit from the tool consumer to the tool provider (which would otherwise be easy to do since the data is passed via the user's browser) and that the request is current and has not been received before. Refer to the OAuth documentation for more details. [OAuth-A] In addition it is recommended that tool providers deliver their services through connections using https to provide additional security of the data being transmitted (which is not encrypted in the request).

2.4 Receiving a launch request

The endpoint URL provided to tool consumers for LTI launches will need to be a script which is capable of processing the request. A typical script would perform the following actions:

1. Ensure all the required LTI parameters are present and have appropriate values: the lti_message_type and lti_version should have their prescribed values, and the resource_link_id and oauth_consumer_key parameters should not be empty.
2. Check that the request comes from a known tool consumer: look up the consumer key to make sure a shared secret is recorded for it.
3. Verify the authenticity of the request: check the timestamp and nonce values, and generate the OAuth signature from the HTTP request and check that it matches the signature value received.
4. Ensure that any parameters which your system depends upon are present in the request: remember that most of the LTI launch parameters are not required by the specification and so whilst the launch request may be valid it may not provide sufficient data for your needs but, wherever possible, such dependencies should be avoided (see below).

If you reach this point it means that you have a valid launch request from a known customer with all the data you require. Hence it is now safe to process the request and provide access to the user. This may involve the following actions:

1. Check if this is the first request from the resource link; if so, create any work area, space or other objects required for activity to take place for launches from this link; if not, update the existing objects with any values which have changed (e.g. the name or description).
2. Check if this is the first request from the user; if so, create any user account needed to enable them to access your system; if not, update any existing account with any details which have changed (e.g. name or email address).
3. Ensure the user has the appropriate permissions for accessing the resource link space; for an existing user this may mean changing their privileges if their role has changed.
4. Establish a login session for the user.
5. Redirect the user to an appropriate "home" page; this may depend upon the resource being accessed and the user's role.

Case Study 1 – WebPA launch

Required parameters:
resource_link_id, user_id, roles

Default parameter values generated for: context_title, resource_link_title, lis_person_name_given, lis_person_name_family

Launch process:

- create new module (or update existing) based on resource_link_id
- create new user account (or update existing) based on user_id
- ensure user account has a type of either "Student" or "Tutor" depending upon the role(s) passed

The precise nature of these actions will depend upon the type of system which is being accessed. Access to content, for example, may not require the creation of user accounts, but may only involve associating the resource link ID with a content item by a user with an appropriate role (such as an instructor). On the other hand, an application like WebPA® [WebPA] would provision a new assessment and user as part of the launch request.

2.5 Identity values

One of the most important tasks in developing an LTI integration is that of selecting appropriate values by which each element can be identified within the tool provider application. The elements which may require a unique identity are:

- contexts;
- resource links; and
- users.

In all cases, the unique ID for an element should be combined with the consumer key to ensure it is unique across a tool provider application which may be connected to more than one tool consumer.

2.5.1 Contexts

In many cases the context from which a launch originated is of no interest to a tool provider – the fact that two resource links are located in the same context does not mean that a user has permissions to access both of these resources, the tool consumer may have additional access rules in place which limit who can see specific links (for example, using the *conditional access* functionality in Moodle to control whether an activity or resource can be accessed by individual users). Hence a launch request should not be used to imply anything about a user's access to any other resource links, even within the same context. However, there may be occasions when the context is of interest; for example, when gathering statistics for teachers. The context is uniquely identified by the `context_id` parameter which has a status of "recommended" in the LTI specification.

2.5.2 Resource links

The `resource_link_id` parameter should be present on every launch and provides a unique ID for the specific link within the tool consumer which was followed by the user. This value may be used to associate the launch request with a resource/activity within the tool provider, either by using the same ID or by implementing a one-to-one mapping between the tool consumer and tool provider IDs.

2.5.3 Users

The user ID value can be as simple as that for a resource link, but this does depend upon how a tool provider enforces the authorisation of users to access resources. An LTI launch request merely indicates to a tool provider that the specified user has the ability to access the specified resource link (with the specified role(s), if any). It makes no statement about the user's authority to access any other resources which may exist for the same tool consumer or context. Thus, when adding LTI support to an application which allows a user to navigate between different resource instances, there is a danger that this may allow an LTI user to use this navigation to access a resource other than the one associated with the link they launched from. To mitigate against this risk it may be appropriate to create user accounts with a different scope level:

a) *Consumer:* combine the consumer key with the `user_id` parameter;
b) *Context:* combine the consumer key with the `context_id` and `user_id` parameters;
c) *Resource link:* combine the consumer key with the `resource_link_id` and `user_id` parameters.

The *Consumer* choice guarantees to provide you with a unique ID for each user within the tool provider system. However, its use does mean that the tool provider will use the same user ID for launches from all the resource links added in the tool consumers. Thus, there is a possibility that users following one resource link, may be able to access content linked from another resource link, thereby bypassing the authorisation process provided by the tool consumer through an LTI launch request. For example, a student may have been unenrolled from a course in the VLE and, therefore, no longer have access to the resource links in that course. If launching from another course allows them to access the end points of these resource links, then the tool provider has effectively provided a back door for users to access resources which they should have been disconnected from. Some solutions to this problem may be available to you. For example, if you control the code for the tool provider application, you can prevent users from switching to a different resource after a launch (this is how the LTI connector for WebPA works). Alternatively, if the tool consumer supports the unofficial memberships service [IMS-C], the list of users for each resource link can be updated on a regular basis (e.g. nightly) so any impact will only be short-lived.

The *Context* choice is similar to the *Consumer* choice (see above) except that it implements an ID value which is unique within a context (e.g. a course). If all resource links within a context are always available to all users enrolled in that context, then there is no issue about providing a back door method for accessing resources. However, if access to some resource links might be restricted to a subset of the enrolled users (e.g. via a conditional release mechanism) then this choice could also provide a way for users to bypass the authorisation implemented by the tool consumer (for which similar solutions as discussed above could be implemented).

The *Resource link* option for creating user ID values is the closest fit to the expectations of the LTI specification. A downside of implementing this choice is that it means that a separate user account will be created in the tool provider for every combination of `user_id` and `resource_link_id` parameters (and for each consumer key). Thus, if a course has 5 links to the tool provider, a user following each of these links will have 5 separate user accounts created in the tool provider. But if you do not have control over the tool provider code then this is a sure way of preventing a user from bypassing the authorisation implemented by the tool consumer - each user will, by definition, only have access to a single resource; accessing other resources must be achieved via a separate user account (accessed by launching from a different resource link in the tool consumer).

Case Study 2 – User scope in WordPress

A case could be made for each of the different user scopes in WordPress, so rather than selecting one at random, the administrator is given four choices when configuring a new tool consumer. The selected option is used as a prefix to the generated consumer key:

- *WP1* – resource
- *WP2* – context
- *WP3* – consumer
- *WP4* – global

So there are choices for how to implement a unique user ID when developing a tool provider LTI integration and that choice will depend largely on the navigation features of the application and an evaluation of the risks associated with users gaining access to other resources from a launch. [ceLTIc-A] But, of course, the choice can be left to the system administrator so that it can be selected to match the needs of a particular tool consumer, as was the case with the LTI connector written for WordPress. [SPVSP-A]

2.5.4 Matching IDs with pre-provisioned data

There are times when a tool provider may already be populated with data related to a tool consumer; for example, course and/or user data. In these situations, there may also be a desire to relate this data with the values received on LTI launch requests. The problem here is that the `context_id` and `user_id` parameters are deliberately intended to be opaque by the LTI specification and are often represented by values such as database keys from the VLE. It is, therefore, unlikely that such values would match with any pre-provisioned data which has more likely originated from a different system, such as a student information/record system (SIS). The best suggestions to offer here are:

a) check whether the tool consumer has any choices over what values are used to populate the `context_id` and `user_id` parameters passed on each launch (in case an alternative choice can be matched with existing data);
b) if supported by the tool consumer, the values of the `lis_course_section_sourcedid` and `lis_person_sourcedid` parameters may provide better matches;
c) LTI 1.2+ includes a custom parameter substitution variable for a user's login ID (`User.username`) as a way of helping to alleviate this situation; the course ID could be entered as a fixed custom parameter value for each launch link created. [IMS-D, Appendix C.1]

There may be a temptation to try matching users on names or email addresses, but this is fraught with problems. The values are not guaranteed to be unique or fixed within the tool consumer, and they may even be editable by the user themselves and so cannot be trusted to be valid. For a user ID, there is an option of prompting the user when they first connect to enter their pre-provisioned credentials so that they can be associated with their LTI identity in the future. Of course, if the matching of user IDs is designed to permit users to access the tool provider via their tool consumer or by logging in directly, then there is the same authorisation issue as described above when opting to use only the `user_id` parameter as the unique ID: users may be given access to resources which they can no longer access via the tool consumer.

2.5.5 LTI and SSO

This issue follows on from the wish to match LTI parameter values with pre-provisioned data. In this case the problem is more specific: is there some way in which a user launching via LTI can be associated with an SSO user account? The use case here is to allow a user to log into both the tool consumer and the tool provider using the same identity provider and be able to access the same resources (notwithstanding that the authorisation issue discussed above remains). A solution for this has been proposed by IMS [IMS-E], though no known implementation currently exists. Its elegance lies in its simplicity. On the assumption that both parties have trust relationships with the same identity provider, the LTI launch is extended to include two additional parameters: the type of SSO used to authenticate the user (e.g. CAS, CoSign, Shibboleth) and the URL of the identity provider. After validating the launch, the tool provider redirects the user to a script protected by the same identity provider. Since the user has already been authenticated, they will fall through this redirection and the tool provider now has access to their SSO identity (e.g. through the `REMOTE_USER` environment variable). The SSO identity can be recorded against the user's LTI identity so that when the user next logs in directly their LTI identity can be looked up and access given to any resources associated with this account.

2.6 Roles

When the `roles` parameter is included in the launch it specifies one or more roles the user has in respect of the resource link. The roles may be taken from the standard vocabularies listed in the LTI specification [IMS-B, Appendix A]; they may also include vendor-specific roles, though these should always be in the form of a full URI.

The most common roles a tool provider can expect to receive are:

- Instructor;
- TeachingAssistant;
- Learner.

Some tool consumers may also support roles of:

- ContentDeveloper;
- Administrator.

A tool provider has the tricky task of mapping the different roles which may be passed by different tool consumers onto its own set of supported roles. Some tool providers may not distinguish between different roles, in which case this area is a non-issue. Where roles are essentially divided between teachers and students, then the Learner role fits well with the latter, whilst the others could be accepted as the former. When a tool provider supports a wider range of roles there can be issues if a similar range is not mirrored in the tool consumer. For example, whilst a VLE may have separate roles for content developers (course builders) and instructors, they may also allow these roles to be held by the same user and, therefore, a tool provider must be able to deliver the functionality relating to both roles in a unified interface. Wherever, possible it makes sense for roles to be dichotomous, especially as different tool consumers may support different roles, so reliance on a specific role is not advised. Most tool consumer implementations of LTI hard-code the mapping between the internal role and the LTI role so the VLE administrator or teacher has little control over the roles which are passed. Only the open source PowerLink for WebCT [SPVSP-B] and the open source building block for Blackboard Learn 9 [SPVSP-C] are known to offer some flexibility in this respect and allow the mapping to be specified on a tool-by-tool basis, thereby allowing them to match the association deemed appropriate to the institution's needs; for example, giving teaching assistants a role in the tool provider which fits with the way in which this role is being used in the VLE. Without such mapping being generally available within tool consumers, any tool provider offering a rich set of roles is well advised to implement a mapping option which can be configured for each tool consumer, either by the tool provider administrator or perhaps by a user having an LTI role of system administrator.

If a user has multiple roles, it is important to check that none of these conflict; that is, when received individually might lead to a different role being assigned to the user. If they do, then a deliberate choice should be made as to which role is to be assigned (assuming that only one role per user is permitted). For example, if a user has both a role of Learner and Instructor (which has been seen in a real launch request from a tool consumer), then the lowest level of privilege is given (Learner). However, this is an arbitrary choice, and the user could equally be given the highest level of privilege. Since the tool consumer is a trusted source of data, there is no clear basis on which to

make either choice where the roles received are inconsistent with those supported within the tool provider.

One final issue to be aware of is that roles may be passed from different vocabularies; for example system, institution and context. Most launch requests are made from within a *CourseSection* and hence most roles received are expected to come from the context vocabulary. However, there are instances where launch requests can be made from outside a *CourseSection* [ceLTIc-B] [ceLTIc-C] and so an appropriate response to these requests should be implemented. Similarly, a user launching from within a *CourseSection*, may have a relevant role from a different vocabulary; the most relevant example being a system administrator. A tool provider may offer an interface to the system administrator from the tool consumer, but it may not be the case that such a user is able to perform a launch request or, if they can, that their role will be passed. In fact there are several possible administrator roles which could conceivable be relevant here; for example:

- urn:lti:sysrole:ims/lis/SysAdmin
- urn:lti:sysrole:ims/lis/SysSupport
- urn:lti:sysrole:ims/lis/Administrator
- urn:lti:instrole:ims/lis/Administrator
- urn:lti:role:ims/lis/Administrator
- urn:lti:role:ims/lis/Administrator/Support
- urn:lti:role:ims/lis/Administrator/SystemAdministrator

The key here is probably to check for any or all of these so that whichever a tool consumer might send is recognised.

There is scope in this area for tool consumers to be more explicit about the LTI roles they support and the mapping used from their internal roles. This would allow tool providers to make more informed choices as to how to respond to roles received in each tool consumer launch.

2.7 CSS

The release of LTI 1.1 introduced a new launch parameter named `launch_presentation_css_url`. This allows a tool consumer to pass the URL of a local CSS file. It does not, as yet, appear to have been widely implemented. There is also potential confusion as to whether it should be a URL to the CSS file for the tool consumer, or a URL to a tool-specific CSS file to apply when the tool provider is launched from that tool consumer. A CSS file is most useful if it overrides style names used by the tool provider to customise the appearance of the application and make it better fit within the local environment. A CSS file consisting of the styles adopted by the tool consumer would be problematic for a tool provider to apply in a simple and meaningful way. However, the LTI connector for WebPA and the open source building block for Blackboard Learn 9 are the only known implementations which support a tool-specific CSS URL. [SPVSP-D]

2.8 Override interface

As well as using CSS to change the behaviour of a tool provider, other parameters may also be used to help tailor an interface to fit specific tool consumers. The following parameters about the tool consumer may be provided on a launch:

- tool_consumer_instance_guid
- tool_consumer_instance_name
- tool_consumer_instance_description
- tool_consumer_instance_url
- tool_consumer_instance_contact_email
- tool_consumer_info_product_family_code
- tool_consumer_info_version

Case Study 3 – Custom parameters supported by WebPA

- logo
- logo_width
- logo_height
- name
- css
- email_help
- email_noreply
- return_menu_text

The last two were added in LTI 1.1 and provide the most convenient mechanism for identifying the type of tool consumer from which a launch request has originated. For example, if the value of the `tool_consumer_info_product_family_code` parameter is "`Blackboard Learn`" or "`learn`" then the launch has come from Blackboard Learn 9, using either the core functionality or the open source building block, respectively. This would allow a tool provider to offer a different experience for Learn 9 users, which may be especially relevant when being opened within a frame or iframe.

It is also possible, though perhaps less likely, to customise the tool provider experience for a specific tool consumer (rather than a family, or brand, of tool consumers). The `tool_consumer_instance_guid` parameter should provide a unique ID for a tool consumer and can be used for this purpose, if required.

2.9 Branding

By using LTI a tool provider can easily offer a multi-tenanted service from a single instance of the application. However, though the use of custom launch parameters, it can be possible to provide some branding of this service for individual tool consumers. For example, these may be to provide an alternative logo, title, contact email address, URL for help, etc. They may also be used as an alternative means of passing values which are not consistently supported by all tool consumers; for example, the LTI connector for WebPA accepts a tool-specific CSS file URL in a custom parameter named `css`. [SPVSP-D]

2.10 Course archive/restore/copy

LTI 1 does not provide a standard mechanism for a tool provider to distinguish between a launch request coming from a brand new resource link and one coming from a link which has been generated by making a copy of an existing one. In some cases this may not be important, but some tool providers may wish to offer teachers the option to also duplicate the content from the existing link. One workaround to this situation is to include a custom parameter in the launch request with a value which can be matched against the `resource_link_id` parameter on launch; if the value matches against a different `resource_link_id` then this would suggest that the launch comes from a copy. Such a solution involves some action on the part of the teacher, unless the resource link is created using the Content-item message process [IMS-F] in which case the tool provider could

automatically add the custom parameter required to record its local reference to the resource/content.

An alternative solution being discussed with the IMS community is to recommend an additional parameter to launch requests which provides a history of the resource link as a list of any previous resource link IDs from which it has been copied. This parameter may have a name of `ext_copy_of_resource_link_id`; each ID would be separated by a comma with the most recent copy first. But, at the time of writing, this is just a topic for discussion with no known implementations.

2.11 Class libraries

Most of the verification of the authenticity of a launch request can be handled by an OAuth library which is available for most programming languages.[OAuth-B] For example, a method to verify a request in Java can be as simple as the following (using the BLTI Sandwich sample code [BLTIS]):

```
public static boolean isValid(String key, String secret) {
  try {
    SimpleOAuthValidator simpleoauthvalidator = new
       SimpleOAuthValidator();
    OAuthConsumer oauthconsumer = new OAuthConsumer("about:blank",
       key, secret, null);
    simpleoauthvalidator.validateMessage(bltimessage.getOAuthMessage(),
       new OAuthAccessor(oauthconsumer));
  } catch (Exception exception) {
    return false;
  }
  return true;
}
```

However, this is just part of the processing of a launch request; other standard tasks include:

- verifying that the request complies with the LTI specification;
- ensure all the parameters required for the tool provider application are included;
- assemble the user, context and resource link data supplied in useful objects; this might include generating default values for missing parameters and assembling values based on how they were sent (such as the user's name which might be passed as separate given and family names, or as a full name, or as both);
- redirecting a user when an error arises with the launch.

Rather than writing your own library to handle this process, why not consider using one which is already available? For example, the ceLTIc project has produced LTI Tool Provider class libraries for PHP [SPVSP-E] and Java [SPVSP-F]. Using and contributing to such open source libraries has the following additional benefits:

- keeps application code separate from the details of the LTI specification;
- simplified validation of launch parameters;
- simpler to support any changes in the LTI specification;
- should help to ensure that applications will pass IMS certification tests;
- support included for LTI services (including unofficial extensions with automatic use of whichever service is supported by the tool consumer where alternatives are available);

In addition, a class library may provide additional functionality to support typical workflows involved in building tool providers. For example, the ceLTIc class libraries include functionality for:

- enabling resource links to be automatically mapped onto a single ID so that users can collaborate in a single space (these resource links may be from the same tool consumer or from different ones);
- allowing consumer keys to be easily enabled or disabled;
- setting start and end times for the validity of a consumer key.

The last of these additions has been used by the ceLTIc project to automate the process of generating credentials (consumer key and shared secrets) on demand for trial/demonstration purposes which automatically expire. [ceLTIc-D]

2.12 Browser issues

LTI is susceptible to all the issues surrounding web-based applications; this section raises some of those most likely to be encountered when using LTI, most relate to using frames or iframes.

2.12.1 Third-party cookies

It is highly likely that the tool consumer and tool provider will be served from different domains and this can be problematic when they are mixed on the same page using a frame or an iframe. This is because tool providers will typically use sessions to remember the state of a user's interaction and this session is identified via a cookie in the HTTP request headers. However, different browsers have different rules and default settings for allowing cookies from different domains (third party cookies) within the same page. If a browser has been set to reject third party cookies, then the session cookie for a tool provider opened in a frame or iframe will not be saved and so the login will fail. Internet Explorer (IE) implements a slightly more complicated solution. By default, for domains in the Internet Zone, IE "blocks third-party cookies that save information that can be used to contact you without your explicit consent". It determines whether a cookie contains personally identifiable information by consulting a policy written using the Platform for Privacy Preferences Project (P3P) protocol. Where appropriate, adding the P3P policy files to the tool provider server and including a P3P header in HTTP response headers can help alleviate this issue. [ceLTIc-E]

Of course, to completely avoid the problem, ensure that a tool provider is opened in a new window (or tab)! Although it is also quite possible for a tool provider to reside on the same server as the tool consumer, again avoiding any cross-domain issues. [ceLTIc-F]

2.12.2 Internet Explorer zones

Internet Explorer (IE) divides the world into Internet, Local Intranet, Trusted Zones, and Restricted Sites. For the majority of users using their own machines most (or all) of the pages they access via IE will be in the Internet zone. However, issues can arise when the contents of a frame (or iframe) are delivered from a server which is in a different zone from the page itself. When this occurs, for example, with WebPA, attempts to download XML reports fail because IE receives the response from WebPA but re-sends the request but without the session cookie header and so it is declined by the WebPA server. This issue only arises when the two servers are in different IE zones.

2.12.3 Support for frames/iframes

There is no requirement in LTI for a tool provider to deliver its content within a frame (or iframe). It may, for example, not make sense given the nature of the application and its UI. The recommended `launch_presentation_document_target` launch parameter should identify where the content requested is to be displayed. A launch request could be rejected (with an appropriate message) if an attempt is made to open the tool provider in a frame; this should be simple for the teacher (or system administrator) to resolve by merely altering the launch settings to select the "new window" option. Alternatively, a tool provider might choose to override the choice by using JavaScript to detect that a launch is occurring within a frame and force the page to be displayed within a new window (or tab). In this case an appropriate message should be displayed within the frame or, if a return URL has been supplied, the frame could be redirected to this URL with a message informing the user that the content should be available in a separate window. Which solution to choose is fairly arbitrary, though the former could be preferred on the basis that it does not usurp the request made by the tool consumer.

Of course, if the parameter is not provided on launch then a JavaScript solution is the only option and it could be seen as being more acceptable to force the display into a new window since the tool consumer failed to note a preference!

3 Issues for System Administrators

This section discusses issues relevant to system administrators for tool consumers seeking to make LTI tool providers available to their users.

3.1 Tool requirements

The addition of an LTI-based tool to a VLE should involve the same type of due diligence process as for any other learning application. This includes:

- checking that its functionality meets needs and is compatible with existing infrastructure;
- agreeing the SLA, licence agreement and any licence fee;
- ensuring it complies with policies such as data privacy.

In order to properly assess the last of these, a system administrator needs to know what data is being passed between the VLE (tool consumer) and the tool. The LTI specification has no required parameters involving personal data, but does support the sharing of user IDs, names, email addresses and roles, as well as data about the link and course from which the launch originated. The tool consumer should have options to turn on and off the passing of such data, but it may not be easy to determine which parameters are required for the successful use of the tool provider, or when the inclusion of a particular parameter may provide enhanced functionality to users. It would be very helpful if the documentation provided by tool providers included the following to assist system administrators in knowing how to configure the link and understand the consequences of their use:

- which launch parameters not required by the LTI specification are required for a successful launch;
- which launch parameters are not required, but their presence gives enhanced functionality to users;

- which LTI roles are supported and what privileges are given to users with each role (this is particularly important if the TeachingAssistant role is supported as it is the one which is most likely to vary in level of privilege given to it by different institutions).

If such information is not readily available, then the best approach would be to start by turning off all the available options for passing context/resource/user data and only turn them on if a launch request fails or to improve the user experience.

3.2 Configuring tools (XML)

The LTI specification does provide a mechanism for defining a connection to a tool provider using XML and some tool consumer implementations provide this option within their UI (e.g. the open source LTI building block for Blackboard Learn 9 and Canvas by Instructure). This XML can be used to specify:

- title;
- description;
- launch URL (including an optional URL for secure connections);
- custom parameters;
- URL for an icon (including an optional URL for secure connections);
- details about the vendor: code, name description, URL, contact email address.

In addition to these details extension sections may be defined for settings specific to individual platforms. For example, an extensions section with a platform name of "learn" is used by the open source LTI building block to enable all the configuration settings to be specified in the XML (e.g. which parameters should be passed, which value to use for a context ID, etc.). [ceLTIc-G] It can, therefore, make it very convenient for customers to be supplied with an XML description of tool providers (or a URL to such an XML file) and can reduce problems with transcription errors. However, not all tool consumers adopt the format provided in the LTI specification; for example, Canvas expects XML with a root node named `cartridge_basiclti_link` rather than `basic_lti_link`. In addition the XML used by Canvas fails to validate because elements are not in the correct order, a new element (`options`) is introduced, and a required element (`vendor`) is omitted. Hence it is not possible, at present, for a single version of an XML description of a tool provider to be defined for use with any tool consumer. So be prepared to request a specific variation for your own needs or edit the XML provided.

3.3 Verifying connections

After configuring a new LTI tool, it is good practice to verify that it works. There is no mechanism in the specification for assisting with this; it is normally a matter of adding the tool to a course and trying a launch. Some tool consumer implementations may provide a launch option from the tool configuration page to make this easier. The main purpose of the test at this point is to verify the URL, key and secret used to configure the tool; if any of these items is entered incorrectly any attempt to launch the tool should fail. (Note that LTI 2 should overcome this issue as the configuration of tools is undertaken as a negotiation between the tool provider and tool consumer servers, without the risk of human error in entering launch parameters.) However, a failed launch request may also occur if:

- the tool provider has not yet set up the details on their system;
- the consumer key has not been enabled by the tool provider;
- the consumer key is only valid for a specific time period and the current time is not within that period;
- the launch request does not include sufficient data to support the requirements of the tool provider;
- the length of a parameter exceeds the tool provider's acceptable limit.

Case Study 4 – Testing mode

Any tool providers built using the PHP or Java LTI Tool Provider class libraries support a custom parameter of "`debug=true`" which causes more detailed error messages to be returned on failed launch requests.

Hopefully, the tool provider returns a detailed reason for any errors which arise as a log entry for the tool consumer, or has an option to display such messages as part of a "testing mode".

3.4 Course archive/restore/copy

When a course containing a link to an LTI tool is copied, for example, the link in the new course will have a different value for the `resource_link_id` parameter when launched. This means that a tool provider will see this as a launch from a new link but without appreciating that it originated from an existing link so that, for example, an option to duplicate the content from the original link could be offered. Tool providers may have a mechanism in place for trying to handle this situation, but system administrators should be aware the any archive/restore or copy actions which take place on the tool consumer system, are not automatically replicated by tool providers and so some manual intervention may be required. This is an important aspect to test when investigating new tool providers.

3.5 Mapping VLE/LTI roles

LTI uses the LIS [IMS-G] vocabulary for user roles. These are likely to be different from those available within a course so a tool consumer will implement some form of mapping from course roles to LTI roles. This mapping need not be on a one-to-one basis, though typically it is. The mapping may be a static implementation applied to all LTI launches for all tool providers, or it may be configurable for each tool provider or even each resource link. The key for a system administrator is to ensure that, as far as they are able, users are given an LTI role appropriate to the tool provider and their course role. If the mapping is static and cannot be altered, then a review of the tool provider to ensure that users are given sufficient privileges within the tool provider and not given privileges which are not appropriate to their course role.

4 Issues for Teachers/Students

This section discusses issues relevant to teachers and students seeking to use LTI tool providers as part of their on-line courses.

4.1 Number of links per course

Typically a link to a tool provider is added to a content area of a course within a tool consumer in a similar way to adding any other form of content. There is no reason for teachers and students to know that a link is actually connecting them to externally-served content; it should form just another

part of a seamless on-line course environment. Whilst the behaviour of following each link is dependent upon the implementation by the tool provider, the normal expectation is that they would act as unique connections to separate resources/activities. Thus, for example, if the tool provider is a quizzing application, adding a new link should add a new quiz to the course. The consumer key and resource link ID passed when a link is launched allow the tool provider to keep each link separate from others added to the same course, in other courses in the same tool consumer, and from those added to any other tool consumer around the globe.

4.2 Pre-populating enrolments and groups

Since the details about the user can be securely passed when they click the link, the tool provider is able to create/update any user accounts required at its end at the time of the launch request. Thus, there is no need for the external system to be pre-provisioned with users; they can be added "on-the-fly". LTI does not, however, provide a mechanism for sharing group memberships defined within the tool consumer. The closest solution is an extension to the unofficial memberships service which includes group information; this has been used for WebPA which, as a peer assessment tool, requires a teacher to have details of all the students so that assessments can be created and each student assigned to a group. [ceLTIc-H]

4.3 Managing outcomes

Many tool providers involve assessable activities and use the Outcomes service to return a grade to the column in the tool consumer's gradebook associated with the link added to a course. Only grade values between 0 and 1 (inclusive) can be passed, but most gradebooks would display this value as a percentage. In some cases teachers (or administrators) have the option to specify the number of points possible so that the grade can be displayed as a score. For example, with a points possible of 60, when a grade of 0.7 is returned it would be displayed to users as 42.

Grades may be returned from a tool provider as a result of an action by the student, or by the teacher, or as part of an automated (possibly regular) process. Once a student has launched a link their cell in the associated gradebook column falls under the control of the tool provider and can be updated at its discretion.

Whilst the LTI specification requires that a grade be transmitted as a value between 0 and 1, it need not represent a score. Some tool providers may associate the values with other meanings. For example, 0 may represent that a student has started an activity with 1 representing completion. The sample Ratings application uses the grade to represent the proportion of the items available which have been rated by the student. [ceLTIc-I]

4.4 Re-using content

Each launch link added to a course is identified by a unique ID. A tool provider will typically use this ID to associate the link with an activity/resource/work space. If a link in a course is copied the copy will have a different ID and hence the association made by the tool provider will be lost. However, some tool providers encapsulate the association within the launch URL (which would, therefore, be unique for each link) or as a custom parameter, in which cases the connection need not be lost by being copied.

4.5 Sharing content

With each link having a unique ID, a tool provider will normally treat them as entirely separate connections. However, there are situations where It could be beneficial for more than one link to be associated with the same activity; for example:

- in order to allow multiple entry points from within a course;
- when an activity involves students from different courses or different tool consumers, even different institutions.

There is nothing specifically within the LTI specification to enable such scenarios, but tool providers may provide a mechanism for doing so, such as a custom parameter. Such functionality has been incorporated into the open source PHP and Java class libraries which will automatically allow additional links to appear to the tool provider as if they have been launched from a primary link. [ceLTIc-J] Check the documentation for the tool provider to see if this functionality is available, as is the case for the LTI connectors for WordPress [SPVSP-A] and WebPA [SPVSP-D].

4.6 Mapping VLE/LTI roles

Most tool consumers allow users to be given a variety of roles within a course, such as:

- course administrator;
- course builder;
- instructor;
- teaching assistant;
- learner;
- guest.

A tool provider may not support such a rich set of roles or, if it does, may not have the same view of the importance of the roles. In particular, a teaching assistant role can vary widely between institutions in terms of the level of responsibility and access they are given to courses. Some tool consumers provide a mechanism for mapping course roles onto LTI roles but, otherwise, the tool provider may allow some control over roles and permissions via a configuration page available to instructors (or administrators). However, in many cases only roles of instructor and learner are supported so there may be issues as to whether course builders, teaching assistants and guests are given the desired level of access.

4.7 "Dummy" users

Many VLEs offer users the ability to view the system as if they had a different role; commonly this is used to allow a teacher to preview their course as a student would see it. If the teacher takes a test in this mode, the VLE may be clever enough to also exclude their marks when generating class statistics. At least in some implementations, this is achieved by generating a new user account which is used during this time.

So what does this mean for LTI connections added to a course. Well, the first thing is that there are no known implementations which actually inform the tool provider that the user who has performed the launch is not a real account. This means that tool providers cannot distinguish their activity from that of real users, hence this may distort the class numbers and statistics. It may also be the case that the `user_id` parameter is not changed, but merely the user's role will have changed from

Instructor to Learner, for example. A tool provider always needs to be vigilant for users changing roles, but in cases like this the change is merely temporary and short-lived. There are, however, some VLEs which do use a different ID for these "dummy" user accounts, however, in the case of Blackboard Learn 9, their recommended practice is for a new account to be created every time the teacher selected the student preview mode. In this case, the number of "dummy" user accounts received by a tool provider could be quite large.

As a teacher, you should be aware that your preview account will not be recognised as such by a tool provider. If this has a side effect on class statistics or licence fees, then it may be advisable to avoid launching LTI connections when using this feature.

5 Issues for Service Providers

This section discusses issues relevant to hosting providers seeking to deliver a service using a tool provider application.

5.1 Service level agreements

One of the benefits of LTI is that is enables applications to be shared across multiple tool consumers. These may belong to the same institution, or be servicing institutions from across the globe. A consequence of this is that, as the service provider, it makes it difficult to predict service levels when the load from other customers can impact the quality of service being delivered. It may take experience of the application and infrastructure which can quickly adapt to changing loads to ensure that customers are satisfied.

5.2 Upgrades

Whilst running a single instance of an application for multiple customers has efficiency gains in terms of maintenance and support, it would normally also mean that every customer must use the same version. Moreover, any move to a new version will be at the same time for all users. Whilst it might be technically possible to build an application such that a single server could continue to run multiple versions, with each customer upgrading independently, the cost of doing so may outweigh the savings of using a multi-tenanted environment and no examples of such an approach are known. In fact, products like Canvas show that institutions are willing to accept cloud-based solutions and have the upgrade cycle dictated to them by the supplier.

5.3 Backups

Since a single instance of the application may be supporting multiple customers, a backup of the application (source code and database) will not easily enable an individual customer to have their state restored to a particular point in time. Nor is it straightforward to provide a customer with a backup to access separately; for example, for performing analytics or when moving to a different service provider. In this case, tool providers should consider implementing import and export functionality for individual tool consumers; this could even be made available to anyone launching with the System Administrator role.

References

[BLTIS] George Kroner, *BLTI Sandwich*, http://projects.oscelot.org/gf/project/blti-sandwich/

[ceLTIc-A] ceLTIc Project, *WordPress and user scope*, March 2013, http://www.celtic-project.org/Project_blog/2013/03/WordPress_and_user_scope

[ceLTIc-B] ceLTIc Project, *Extending LTI functionality in Learn 9*, September 2012, http://www.celtic-project.org/Project_blog/2012/09/Extending_LTI_functionality_in

[ceLTIc-C] ceLTIc Project, *WebPA dashboard*, September 2012, http://www.celtic-project.org/Project_blog/2012/09/WebPA_dashboard

[ceLTIc-D] ceLTIc Project, *Evaluating apps: the quick way with LTI*, April 2013, http://www.celtic-project.org/Project_blog/2013/04/Evaluating_apps_the_quick_way

[ceLTIc-E] ceLTIc Project, *A Learning Experience – P3P and Cookie Blocking*, November 2012, http://www.celtic-project.org/Project_blog/2012/11/A_Learning_Experience_P3P_and

[ceLTIc-F] ceLTIc Project, *Parasitic tool providers*, January 2013, http://www.celtic-project.org/Project_blog/2013/01/Parasitic_tool_providers

[ceLTIc-G] ceLTIc Project, *Using an XML tool descriptor*, January 2013, http://www.celtic-project.org/Project_blog/2013/01/Using_an_XML_tool_descriptor

[ceLTIc-H] ceLTIc Project, *Synchronising group data*, December 2012, http://www.celtic-project.org/Project_blog/2012/12/Synchronising_group_data

[ceLTIc-I] ceLTIc Project, *Rating with Outcomes*, June 2013, http://www.celtic-project.org/Project_blog/2013/06/Ratings_with_Outcomes

[ceLTIc-J] ceLTIc Project, *Using LTI to enable collaboration*, November 2011, http://www.celtic-project.org/Project_blog/2011/11/Using_LTI_to_enable

[IMS-A] IMS Global Learning Consortium, *Learning Tools Interoperability*, http://www.imsglobal.org/lti/

[IMS-B] IMS Global Learning Consortium, *Learning Tools Interoperability Implementation Guide, Final Version 1.1.1*, http://www.imsglobal.org/LTI/v1p1p1/ltiIMGv1p1p1.html

[IMS-C] IMS Global Learning Consortium, *Context memberships service*, http://developers.imsglobal.org/ext_membership.html

[IMS-D] IMS Global Learning Consortium, *Learning Tools Interoperability Implementation Guide, Public Draft Version 1.2*, http://www.imsglobal.org/lti/ltiv1p2pd/ltiIMGv1p2pd.html

[IMS-E] Charles Severance and Stephen P Vickers, IMS Global Learning Consortium, *LTI, SAML, and Federated ID – Oh My!*, http://developers.imsglobal.org/LI2012-lti-saml.pdf

[IMS-F] IMS Global Learning Consortium, *Content-Item Message*, http://www.imsglobal.org/lti/ltiv1p2pd/ltiCIMv1p0pd.html

[IMS-G] IMS Global Learning Consortium, *Learning Information Services*, http://www.imsglobal.org/lis/

[OAuth-A] IETF, *OAuth 1.0 Protocol*, http://tools.ietf.org/html/rfc5849

[OAuth-B] OAuth, *Code*, http://oauth.net/code/

[SPVSP-A] SPV Software Products, *LTI Connector for WordPress*, http://www.spvsoftwareproducts.com/php/wordpress-lti/

[SPVSP-B] SPV Software Products, *BasicLTI PowerLink*, http://www.spvsoftwareproducts.com/powerlinks/basiclti/

[SPVSP-C] SPV Software Products, *BasicLTI Building Block*, http://www.spvsoftwareproducts.com/bb/basiclti/

[SPVSP-D] SPV Software Products, *LTI Connector for WebPA 2*, http://www.spvsoftwareproducts.com/php/webpa-lti/

[SPVSP-E] SPV Software Products, *PHP LTI Tool Provider class*, http://www.spvsoftwareproducts.com/php/lti_tool_provider/

[SPVSP-F] SPV Software Products, *Java LTI Tool Provider package*, http://www.spvsoftwareproducts.com/java/lti_tool_provider/

[WebPA] Centre for Engineering and Design Education, Loughborough University, *WebPA (on-line peer assessment tool)*, http://webpaproject.lboro.ac.uk/