Incorporating a TinCan LRS into an LMS

If you are not familiar with Project Tin Can, the TinCan API, and the term LRS, please refer first to the Tin Can API specification.

Since an LRS is meant to be a component with limited but well defined capabilities, we expect that in many cases it will be desirable to integrate an LRS into an LMS, and that it will be beneficial to define expected behaviors for such an LMS. This document was created to define those expected behaviors without cluttering the core LRS documentation.

In particular, content packaging, launch, and import should be defined for an LMS, but not for an LRS -- it just stores and retrieves learning records, not content.

Packaging

Content, activity definitions, and activity provider definitions may be packaged for TinCan. The primary object being packaged is always activity definitions. It is valid to create a package that contains only activity definitions, however content and activity provider definitions may only be packaged with an associated activity definition.

A TinCan package must always include a TinCan metadata file, this is a file named “tincan.xml”, conforming to the TinCan schema. If not including content, this file itself may be used as the TinCan package. If content is to be included, then this file must be placed into a zip file with the content.

A TinCan package must contain exactly one “tincan.xml” file. The location of the “tincan.xml” file is considered the “root” of the package. All files within the package should be under the root. So, it is valid for tincan.xml to be in a nested directory structure within the zip, but only if every directory above it contains nothing but a single sub-directory.

If HTML files are included in the content package, they may link to each other, or other resources within the package, using relative paths based on the package structure. They may also link to external resources using absolute paths.

Activity Definition

The activity definition section of tincan.xml maps to the activity definition defined in the TCAPI document, with the exception of “launch” and “resource”.

Launch is an absolute or relative path to a document which can be launched by the LMS in a web browser in order to “deliver” this activity.

Resource is an absolute or relative path to something that a person can either read or download which will enable them to experience this activity, for example: an application to run, a video to view, or directions to a physical location for an event.

Only one activity definition within a package may contain launch or resource elements. This limitation should be lifted in future TinCan versions but the implementation consequences of multiple launchable activities per package need more thought first.

NOTE: Activities do not have a hierarchy, and are declared as a flat list. Any hierarchical context must be reported by the activity provider, using the “context” portion of each statement. (if that context is to be preserved). If emulating a traditional SCORM package using TinCan, consider adding a “grouping” activity to each statement, which corresponds to your “root of the activity tree”, and also a “parent” activity.

Activity Provider Definition, activity groups.

The tincan.xml file includes one or more groups of activities. The reason for breaking up activities into groups is to enable defining different authorized activity providers for each group. Each group of activities has a “provider” section, which MAY be used to declare to the LMS/LRS what applications (activity providers) should be allowed to report statements about each group of activities. The elements in this section reflect the information that must be registered duiring the OAuth application registration process, and provide a way to get this information into an LMS without the administrator having to go through a registration UI.

The schema for TinCan.xml is: http://projecttincan.com/tincan.xsd

TinCan.xml doesn't have to have all activity IDs that will be used in it, what it should have is:

1) The activity ID that is considered the root activity for this package, along with a launch link, so the LMS knows what to launch.

2) Any activity details (such as activity descriptions) that should be available to reporting systems, but will not be (or may not be) sent by the activity provider when reporting statements. That is, TinCan.xml may be used to describe activities to the LRS, as an alternative to doing that description at runtime.

Import

When importing a TinCan package, all the content, activity definitions, and activity provider definitions in the package will be imported.

If any activity definitions are loaded then all the files within the package starting from the “root”, and excluding the file “tincan.xml”, must be placed in an accessible location by the LMS. For any activity definitions that have “launch” or “resource” defined, the LMS will store those values

as activity profile entries in its associated LRS. Any relative URLs will be transformed into absolute URLs based on the location the LRS stored the content.

Language / Internationalization

Many of the entries in the TinCan schema have an xml:lang attribute. Where the language of the associated entry is known, it should be declared. Where an entry is available in multiple languages, it should be repeated for each language. If the language is unknown, then the attribute should be left blank.

Launch

Tin Can APs do not need to be launched from an LMS, however it is still an option. When an LMS launches a Tin Can AP, it will provide the necessary information for that AP to track back to the LRS (endpoint, learner information, credentials, and optionally registration, activity ID, platform,language, and grouping). The format of the launch URL will be as follows:

<AP URL>/**?endpoint=**<lrs endpoint>**&auth=**<token>**&actor=**<learner>[**&registration=<registration>**] [**&activity_id=<activity ID**>[**&activity_platform=<platform>**][**&Accept-Language=**<accept language>][**&grouping=**<grouping activity ID>]

Note that that some of the parameter values include reserved characters, and even other URLS, and therefore must be URL encoded.

Example launch link (shown without URL encoding and with line breaks for readability):

http://example.scorm.com/TCActivityProvider/
?endpoint=http://example.scorm.com/lrs/
&auth=OjFjMGY4NTYxNzUwOGI4YWY0NjFkNzU5MWUxMzE1ZGQ1
&actor={ "name" : ["Project Tin Can"], "mbox" : ["mailto:tincan@scorm.com"] }
&registration=760e3480-ba55-4991-94b0-01820dbd23a2
&activity_id=http://example.scorm.com/tincan/example/simplestatement

Partial launch information may also be provided by an LMS in the form of a launch link, which may consist of only endpoint information, or may include learner information but not credentials. In this case, the AP would have to have been configured with or prompt for the necessary information.

The LMS should specify Accept-Language, if it knows the learner's language preferences. Except for its location in the query string instead of the header, Accept-Language should be constructed and interpreted according to RFC 2616 (HTTP 1.1).

The LMS should specify a grouping activity if the activity being launched is considered, by the LMS, to be part of a larger group of activities for reporting purposes. The grouping activity specified should then be added to the context of each statement the activity being launch reports, so that these statements can later be identified as all being related to the grouping activity.

The LMS should specify a registration on a launch link if the launch is logically part of an LMS "registration". The Activity Provider will then store the specified registration in context when reporting statements.

Any parameters that are not defined here which are passed on a launch link should be echoed back to the LRS when reporting statements associated with that launch. This allows an LMS that is integrated with an LRS to pass through additional context information.

If no launch information is provided, then the AP must minimally be configured with the LRS endpoint it should track to. The AP may also be configured with credentials from the LRS, in which case credentials need not be obtained for each learner.

If launch refers to an activity with associated protected content, the launch link will include additional parameters to support access to that protected content.

OAuth

If the activity being launched has an associated registered OAuth application with the LMS, the LMS should not include an "auth" parameter in the launch link. The Activity Provider / OAuth application is expected to authenticate using OAuth, which may involve asking the learner to re-authenticate.

Other Scenarios

The process of getting launch information from an LMS to an AP in a manner other than a launch link (URL) is not defined. Although it is a goal of the TCAPI to support out of browser scenarios, this is supported by allowing the AP to pass information to a LRS about learners and activities that have not been previously defined in the LRS. That is, out of browser scenarios are supported by removing the requirement for the LRS to launch the activity. Minimally, the AP must be configured with the LRS endpoint, and usually will also need authentication credentials.

Private Content Access and Tin Can

This document describes a companion specification to the Tin Can API for the purpose of gaining access to content that is stored on an LMS, but which requires authentication to retrieve. This is needed since even though Tin Can allows tracking of experiences for which the content is not stored on an LMS, or for which there is no traditional content, the LMS is still

a convenient place to store the content associated with a learning experience. Since Tin Can does not require an active browser session, the content may no longer be retrieved by relying on that session.

Since the problem is accessing content which is stored on an LMS, and to avoid the need for a complex permissions scheme, this access method will apply only to content that has been uploaded in a Tin Can package, and only where the accessor of that content has been launched using a Tin Can launch link.

In addition to the launch link parameters described in: "Incorporating a TinCan (0.9) LRS into an LMS", the following parameters will be provided as needed:

> **content_token:** The authorization token to be included on requests to the content endpoint. If missing, the content is not protected and must be accessed directly.
>
> **content_endpoint:** the absolute URL to use to access protected content. If content token is specified, but the endpoint is not, the TCAPI endpoint, with the postfix "content/" is to be used as the content endpoint.

When a client application needs to access a protected resource, it will first determine the path to the protected resource based on the content endpoint, and the relative path of the protected resource within the Tin Can package.

For example, in the Golf Example Tin Can package, there is a resource 'Etiquette/distracting.jpg'. If the content endpoint is: https://example.com/TCAPI/content/, then the following path would be built to access this protected resource, and the specified content_token is added to that URL. It should not be necessary to merge the content token with other query string parameters, as even though content may use query string parameters when loaded to determine how to behave, they should not be necessary for retrieving a resource from the server. Note that this methodology works for retrieving 1 resource at a time, an HTML document loaded in this way which includes relative paths would have broken links.

https://example.com/TCAPI/content/Etiquette/distracting.jpg?content_token=b50607fb-956e-429f-b89e-388c43dbbbcf

The client may then issue an HTTP GET to this URL to access the resource.

In order to retrieve an entire content package, the content endpoint would be combined with the content token, with no additional path information. So the following URL would retrieve the entire content package, in zip format: **https://example.com/TCAPI/content/**?content_token=b50607fb-956e-429f-b89e-388c43dbbbcf

The server will not enforce any authentication scheme on the content endpoint, except validation of the content token. The content token will serve both the validate the request, and

look up what Tin Can package the request is associated with. The content token will remain valid as long as the ‘auth’ parameter sent in the launch link remains valid.

Upon receiving a request on the content endpoint, and validating the content token, it is recommended that the server issue a 301 redirect to the URL on the LMS where the content may be accessed, and include on the URL a signature that the LMS will recognize to allow access to the content.

If the above behavior is not practical, the server must instead handle GET, HEAD, and OPTIONS requests according to the HTTP 1.1 specification, as if the server was hosting the content at the specified URL. Note that this includes properly supporting headers such as ETag, Range, if-*, etc.

The behavior described above is intended to allow clients to access individual protected resources without implementing any Tin Can behavior. For example, this is important in order to enable handing off a URL to a video player, that player needs only support HTTP, it does not need to have any Tin Can specific logic. It will be still be necessary for the Tin Can client application to generate the correct URL for each protected resource. So a resource which when loaded then refers to other resources (such as a web page), must be loaded by a client with additional logic to fix those links.