Reference document

Openlynk is a combination technical specification and light agreement to create connexions between Ed Tech companies' platforms.

The main goal is to improve the experience for students when content or services from a provider platform is linked to from a host platform, improve the ease with which teachers can create these links and ultimately automate these kinds of links.

By no means does Openlynk imply how these links are to be monetized, nor imply that data from one provider is automatically available to others. This is about collaborating on a format and modus operandi, not on the business aspects and each partner is responsible to allow or not any of the implementations.

This document attempts to define the different ways these various platforms can interconnect, although this is by no means an exhaustive list.

Some combination of tools may not fit the model directly, but Openlynk can still be used to demonstrate an intent to collaborate by various companies.

Some integrations described here will start off as very Studyo-centric for practical reasons and to gain time, simply because this is the first host platform to support the various types of interactions. As the standard evolves, we will try to dissociate Studyo from the equation as much as possible and encourage other tool developers and content developers to tie-in using the same protocols to encourage interoperability.

Definitions

Host
Refers to the app or environment which calls upon an external resource to integrate a provided link within its own resources. i.e. Studyo would be a host for a resource hosted on J'accorde.

Provider

Refers to the app or environment which provides information to the host app. This can be in the form of a list of resources and links, a full web interface with actions based on the resulting user interactions or backend-provided information.

Resource picker
The way the host can select the external resource from the provider. This should normally be provided by the provider, but simple implementations could build the content picker internally based on the appropriate data shared by the provider.

Integration types

Pasted URLs with auto-format

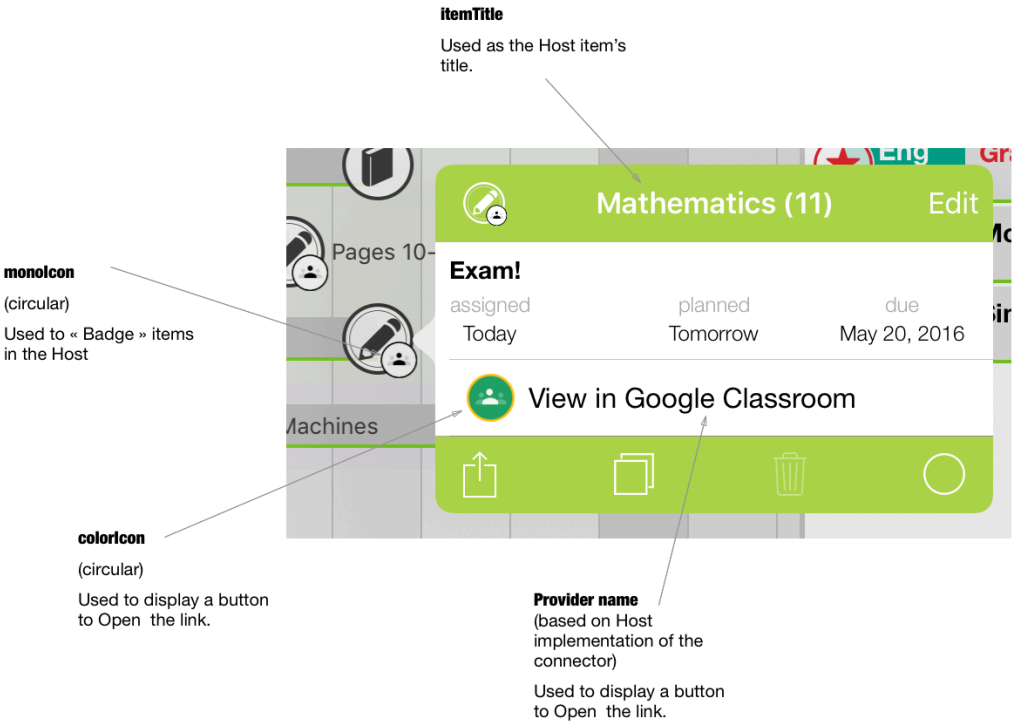
As a minimalistic type of integration, the host environment can opt to simply automatically display pasted links using a custom icon. This requires no work from the Provider, except perhaps provide black & white and color icons which respect their marketing guidelines.
This could be a way to publicly indicate a platform supports the Openlynk standard, because they are engaged in the process and plan to provide better options in the future as time and resources permit. Yet, even if the user experience for students is good, this is the least useful for teachers who must copy and paste links instead of using a picker.

Adopting Openlynk at this level simply indicates that a provider supports the use of their logo and identity and endorse the Openlynk project. They may or may not implement deeper integrations in the future.

[Illustration of Add resource > Paste URL > URL swapped with representation]

Host built-in implementation
The host must obviously define which providers it supports, allow for a place for links to provider resources to be pasted and know what to do to interpret and display the pasted links.
This can be done by integrating the appropriate resources in the host itself, such as a black and white or color icon or button to display an enhanced link.
But the host may be able to infer some of the information if the provider provides some metadata on their end.

Here is an example of what Smart URLs look like once in place:



Openlynk meta-tags

If a provider wishes to personalise or have better control over the experience, they can add some meta-tags to their pages to make it easier for the host to build the link to a resource. The meta-tags will ensure that when pasting a link to a page, the host can get the most recent version of the appropriate icon representation, a proper title and the precise URL required to link to the resource.

Example: Say a host is implementing a personalized link to a provider's resources, it can build it all itself, detecting that the url being pasted contains a specific domain for example, and use its own version of an icon to display the link. If the provider provides metatags pointing to specific versions of it's icon and a better version of the link, it can place these in the source page. The host will then use these meta-tags to build the final link and image.

See the technical reference for the list of required and optional meta-tags

Openlynk Resource Pickers

Resource pickers allow the host to provide teachers with a better experience to select resources and receive a URL pointing to these resources. Once the resource link has been acquired, the rest works exactly as it does for Pasted URLs.

The look and feel of the picker will vary with the type of content, and implemented by the provider so it reflects the branding of the resource and to provide the best experience to the user.

The picker must allow the user to authenticate if required and navigate the user's content in as efficient way as possible. The picker's size should be limited and not present all the info normally found on a site. Ideally, a clean login page adapted to mobile is best. Navigation should be simple and be supported on mobile platforms as well and each resource must be selectable with an "Add" or "Select" button.

If the provider only requires a link to the generic site to be used, then it is probably best to implement it directly in the host. I.e. the resource is a calculator and does not provide specific resources, but only one resource to all. Same for a serious game or link to an app itself.

[Items below this point are not yet technically documented]

Host pickers

In this scenario, the provider pushes information to the host from its own user interface. One example could be an online tool which wants to schedule a task in a host calendaring or planner.
This process is manually triggered by a teacher wishing to add something to their host program and the UI to perform this is provided by the host environment.

This can be done on iOS as well as the web using the following means:

Native iOS Extension (iOS provider, iOS host)
When running on an iPad, the host app should provide an Extension to give access to the destination container (where the link will be stored in the host). The host thus provides an Extension which displays a form of picker appropriate for it's data (i.e. Studyo displays the ability to create a new task or add to an existing task).

When invoking the Extension (by supporting the iOS Share Sheet), the provider must supply at least the link to the resource being pushed to the host (i.e. not a document itself, but a link which would open the document were a student to open the link). So as an example, in Google Classroom, instead of sharing the content of an assignment to other apps, it should also provide a link to the assignment within Google Classroom.

Basically, this should be the same information that is described above for the Resource picker. The host then takes this information,and uses it to add a link to its own object (i.e. a Studyo Task).

[Insert implementation details]

Deep Link
If the provider has an iOS app, it should provide host providers with the scheme of these links for a better experience. The deep link is what should be provided in the payload, as well as a generic web link. The host must find the best link to open depending on the context. For this to work well, the provider should make hosts aware of their URL scheme so the host can detect if this can be opened with an app on the iOS device (Apple rules)

Web picker
If the provider is a web platform, it obviously doesn't have access to the iOS Extensions. The host must then provide a web-based picker equivalent.

[Insert implementation details]

Advanced automated implementations

Openlynk publishing protocol

The general concept of the publishing protocol is that a provider could have the ability to add content to the provider directly from its back-end, without the host having to do anything except provide an API.

It is as of yet unclear exactly what this protocol could look like or how generic it can be at this time.

Technical details forthcoming.

Openlynk fetching protocol

When a provider implements the Openlynk fetching protocol, it can provide a list of resources available to the host via its API. For example, a tool could fetch a specific occurrence of a resource to get its properties, or get a list of all resources associated with a group or user.

Additionally, the provider should implement a few extra features to let the host know about changes proactively, to avoid polling. For example, if a resource is deleted, updated, or changed

in a way which can affect the host (due date, link, notes), it should post a notification to the host at the appropriate endpoint.

If the provider does not provide a notification system, the host should then poll the provider at regular intervals, unless the provider indicated that the resource requires no polling.

An extra level of integration would allow to modify information for specific resources, such as state information and allow the host to change its state.

[Enter implementation details here]

Other Openlynk implementation possibilities

Some platforms do not offer reserces in the same way, but offer services which may apply at various levels.

For example, a tutoring service like GradeSlam or a communication platform such as Prollster might offer something which is not a URL or which should be presented in some unique way.

The details of such implementations has to be done on a more platform to platform basis, but are considered as part of Openlynk to allow for a general message of collaboration to be shared between all the companies involved.

List of companies which have agreed to participate

Studyo
Netmath
ChallengeU
Classcraft
GradeSlam
Affordance
Desmos
PermissionClick
Educathèque
Edu-Techno
Scoop!
Prollster
Peetch
J’accorde