Apollo Innovation

This technical whitepaper demystifying the learning platform that Apollo Tech has built.

Shridhar Navanageri and the entire platform team

Apollo Innovation Learning Platform Overview

Motive

The education industry is going through disruptive change, a revolution is taking place. For a revolution to take place in education, changes will have to occur, that will disrupt the status quo.

According to Clayton Christensen, Michael Horn and Curtis Johnson writers of *Disrupting Class: How Disruptive Innovation Will Change the Way the World Learns*, the key to revolutionizing the classroom is not just by adding technology, but rather by the ways that technology will be introduced. They believe that future schools must be student centric.

Because students have different types of intelligences, they do not all learn the same way. To serve all students effectively, learning should be both personalized and customized. Under the current system, customizing education is expensive. Disrupting Class recommends introducing more computers based learning disruptively. "We need to introduce the innovation disruptively--not by using it to compete against the existing paradigm and serve existing customers, but to target those who are not being served -- people we call non-consumers. That way, all the new approach has to do is be better than the alternative -- which is nothing at all. " (Christensen, Horn, Edutopia 2008).

Hence, there is a need for a learning platform that breaks the paradigm of one-size fits all model, instead focuses on individualization, personalization.

Introduction

Apollo Learning Platform (ALP) is a cloud based multi-tenant education system, composed mainly of an open source stack that incorporates the Service Oriented Architecture (SOA) to provide virtual access to classes, class content, grades, assessments, and other external content, with data embedded in every aspect. ALP provides capability to access the applications across multiple devices and consumption platforms in a consistent way, could it be a browser based application, mobile app or a tablet app.

ALP consists of applications that allow the full life cycle management for a course (class). The course originates from the instructional design developers (IDD) and gets handed off for the instructor to customize where applicable, permissible and relevant, before eventually getting surfaced to the learner. The system allows individualization and customization of the syllabus at each level*. The system also supports handling the scenarios where the instructor gets assigned at a later time, through a mechanism known as auto-publishing.

ALP is built using an integrated set of interactive online services, rich user experience through simplistic design, interfaces that is delivered through a thin

client model and includes social tools known as Personal Efficiency Tools (PETs) that provide instructional designers, content creators, instructors, learners and others involved in education with information, tools and resources to support and enhance educational delivery and management.

One of the key distinguished concepts behind the ALP design is Adaptive Learning Experience (ALE). The learning platform utilizes traditional data/analytics as well as big-data (using Hadoop and toolset) to provide personalization capabilities for students as well as providing faculty and instructional designers with critical information to drive better engagement and outcomes.

At the core, these are the major components:

- The authoring capabilities for a course and content creation, usage; for the instructional designer, instructor.
- The runtime capability to seamlessly integrate with content and display them in a unified and pleasant way to the learner, with a seamless integration capability to Independent Software Vendors (ISVs).
- The personalization aspect that allows serving individualized and personalized course materials that enable the learner to reach the goal faster, while helping them to meet the objectives/goals set for the course.

Architectural vision

The market leading design firm, IDEO (http://www.ideo.com), provided the User Experience (UX) ideas by conducting several user experience tests across different places, different demographics, different age group etc. This means that the market needs were analyzed and understood, before starting to design the platform.

There were quite a few options, but the reason for building the platform from ground up was driven by the needs, facts, challenges and requirements presented by the industry at the time.

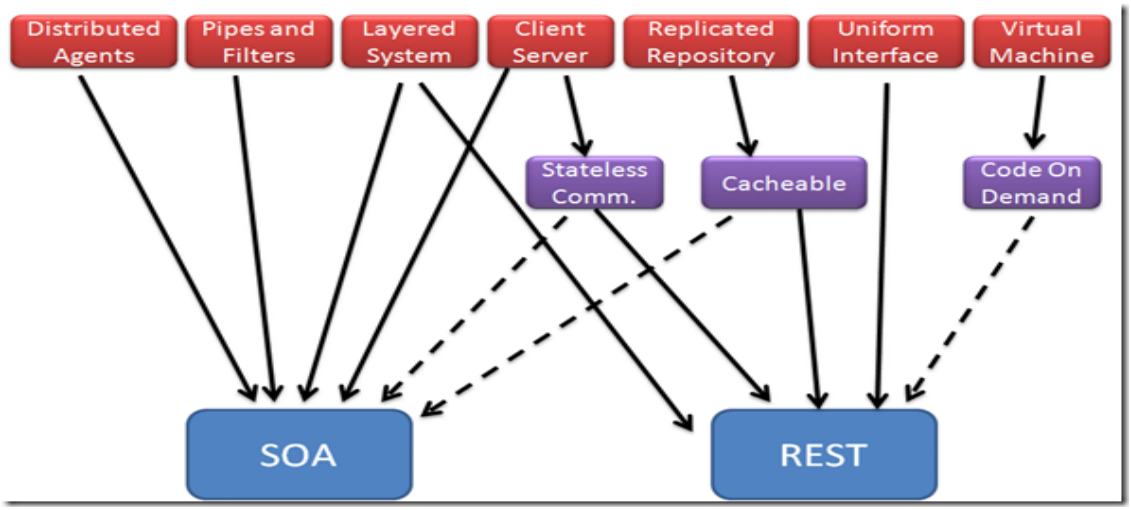
- We needed a system that would allow creating or authoring a course empowering the instructional designers to define the templates for a course.
- We needed a system that would allow the instructor the ability to customize the course template by virtue of allowing them to teach the course more efficiently, while honoring the authorities set by the institution or the instructional designer.
- We needed to develop applications with simple, unified user experience, yet focused on meeting the objectives in the course for the learner.
- We needed to build a platform that will analyze the cognitive DNA of a student and allow applications to be built on top of this data by analyzing the collected data and inject the remedies or relevant content back to the learner.
- We needed a system that will unify the metadata across the system and allow deriving the meaning based on the input from various systems involved.

Core principle

The platform was set out with a mission to build a composite system that would encompass a learning platform, which breaks away from the monolithic application-building paradigm. The platform prompted and promoted the need to build decoupled components across, could it be a horizontal service, vertical service or an application. Each component is designed to be re-used, re-purposed. The platform is built with primitives that allow building innumerable things.

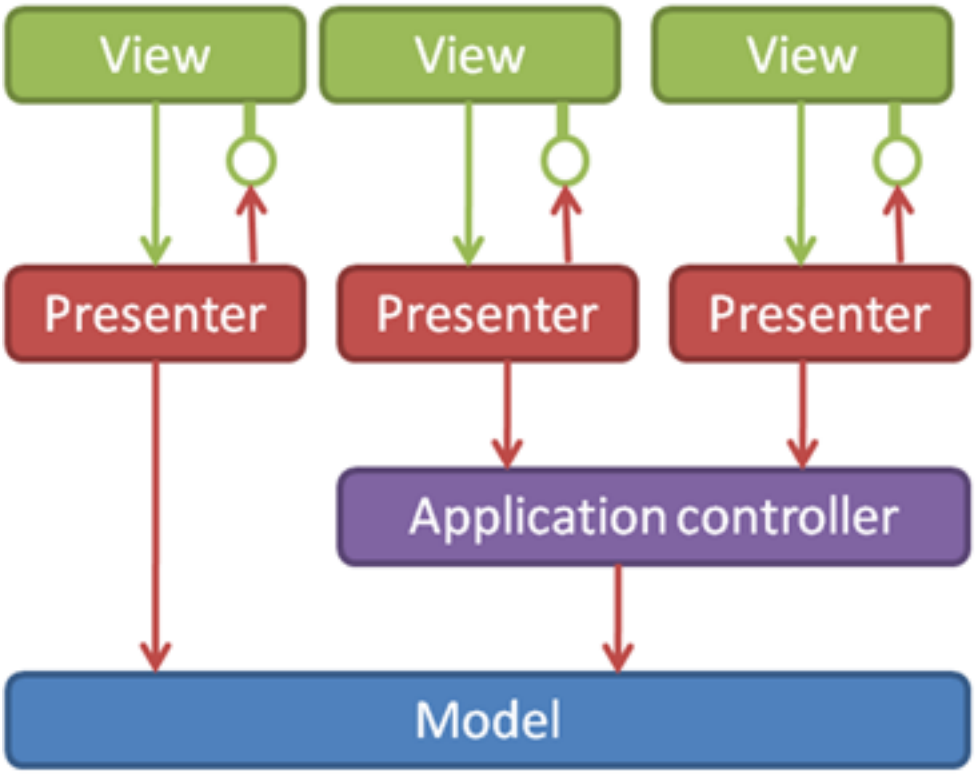
Due to the principles, the Service Oriented Architecture (SOA) is adapted. As it is widely known, some of the advantages of adapting SOA architecture are:

- Abstraction.
- Autonomy.
- Composition.
- Discoverability.
- Formal contract.
- Loose coupling.
- Reusability.
- Statelessness.
- Configuration flexibility.
- Improved reliability.
- Ability to scale operation to meet different demand levels.



The client application development framework is built to compliment these capabilities on the application side, as well. The framework allows the developers to focus on the application development, while doing the heavy lifting of the core concepts through code generation or similar mechanism. The client application development framework promoted the ability of the client applications to make use of the client machine computing power through which the applications provide

much more responsive, faster navigation, deep linking, bookmarking and other Rich Internet Application (RIA) capability.

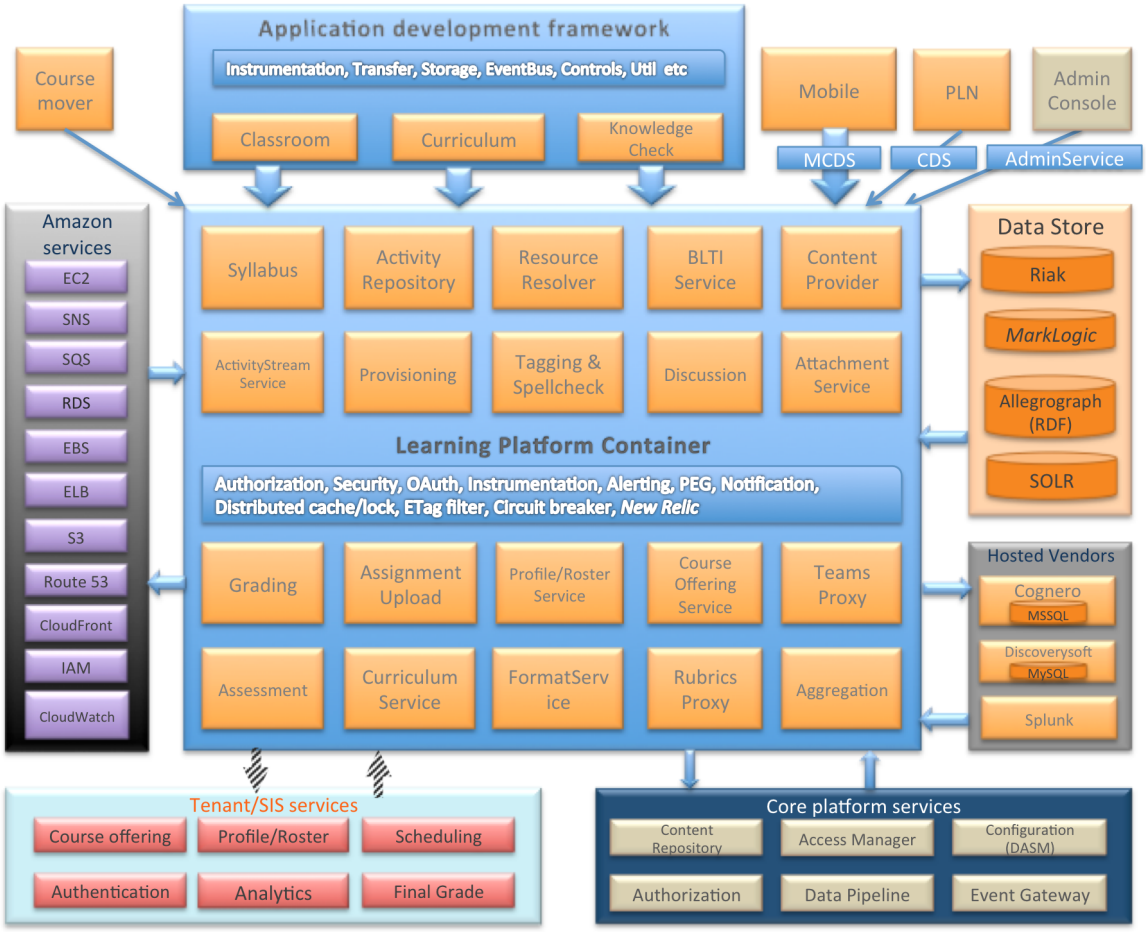


The architecture calls for cleaner separation between the services development and client side development, allowing complete decoupling between them. This allows teams to develop features parallel and integrate with each other at different times.

The architectural design decisions were made based on these principles.

- The platform container supports OSGi based service development or a regular web application driven development.
- The persistence layer allows expanding and scaling through the usage of key-value store (Riak) or Relational Data Store (RDS), where applicable.
- The application development framework supports ReST based clear separation and follows same build, continuous integration and deployment mechanism.

The component architecture for the learning platform is as shown below:



The ALP consists of composite structure built with several individual blocks. This could be compared to a structure built using the LEGO blocks. Different kinds of blocks from standard block sets can be used for constructing endless variety of structures. However, in reality, few components in the complex component structure expect other components providing specific functionality through the declaration of runtime API dependency; they need to observe system-wide policies for security, flow-control, overload control, fault detection and handling; they do on infrastructure for communication, coordination, state maintenance, execution tracing, etc.

The platform consists of learning specific horizontal services, platform services, data store, data pipeline and applications amongst few key components. It has a soft dependency on the tenant services such as offering, profile, roster, scheduling, authentication, analytics, outcome (final grade) services. All of the dependencies are well defined through schemas and hence easily pluggable for another tenant.

The services are built on top of the learning platform container that provides many of the capabilities such as authorization, security, instrumentation, alerting, circuit breaker, ETag, distributed cache/lock and others, out of the box. All of these

components are configurable through configuration or dynamic injection. This way the common infrastructure and shared level services are already available for each service and they can focus on the specific service related logic.

Similarly, the applications are created on top of application development framework that provides the infrastructure level components such as instrumentation, transport, history, evening and other additional capabilities out of the box. Most of these capabilities are available to the applications developers through the usage of code generation concepts.

Data as the platform

As one would expect, the core competency of the ALP is the ability to deal with humongous amount of data and yet derive meaning or recognize the patterns out of the data. The data platform allows multiple variations to the consumption, starting from near real time availability of data to access to raw data, to processed data in tabular format. The same data produced by the applications and services are used for a variety of purposes:

- ✓ Personalization
- ✓ Insights
  - Business
  - Faculty
  - Students
- ✓ Profiling/Metrics.
- ✓ Operations.
- ✓ Research.
- ✓ And many more…

Capturing the click stream from the application plays an extremely important role in many ways, considering the applications are developed using RIA frameworks. However, the instrumentation on the application side is implemented carefully so as not to interrupt or affect the user experience. The client side instrumentation is implemented using a rule engine that computes the best scenario to submit the data in a batch to the capture service, without impacting the application flow. Considering the client browsers or machines are located on different time zone, the time stamp applied on the capture event is stamped using a server time (based off the Lamport - http://en.wikipedia.org/wiki/Lamport_timestamps algorithm derivation).

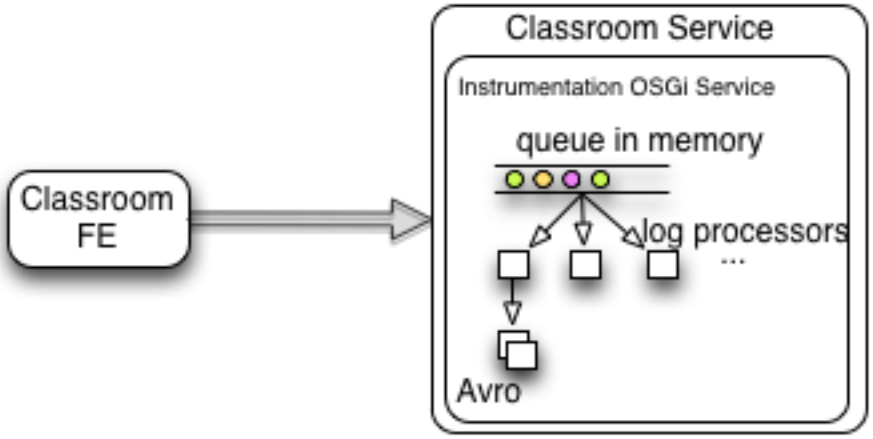
Capturing just the client side click stream doesn't provide a holistic view, until the same implementation is incorporated on the service and data layer as well. Hence, the instrumentation is incorporated as the core platform capability, that all the services could readily make use of. The same time stamp is being used while stamping the time on each of the server event as well. Hence, once after merging the events and ordering them chronologically, suddenly opens up a whole new opportunity to not only see the holistic view of flow, but also allows the system level analysis to even see which component is taking a time while processing the request.

This mechanism also allows a certain module to change the level of logging using external configuration so that in case of a debugging session, the technical support could replay the error condition while a lot more verbose input is being produced.

Both the client side and service side instrumentation allows a plugin model, where the same data could be transformed and/or filtered and submitted into a different end point/system. If the plugin sends the data out to external vendors then the confidential information such as profile gets masked that only the platform could unmask. This is a very important feature to keep in mind from the external system point of view, since it opens up a whole lot of use-cases for a tenant system to build variety of tools, dashboards and applications on top of this data.

E.g. This mechanism is being used to deliver the ARA event in UOPx (University of Phoenix).

The processing of the data is simplistically shown as below:

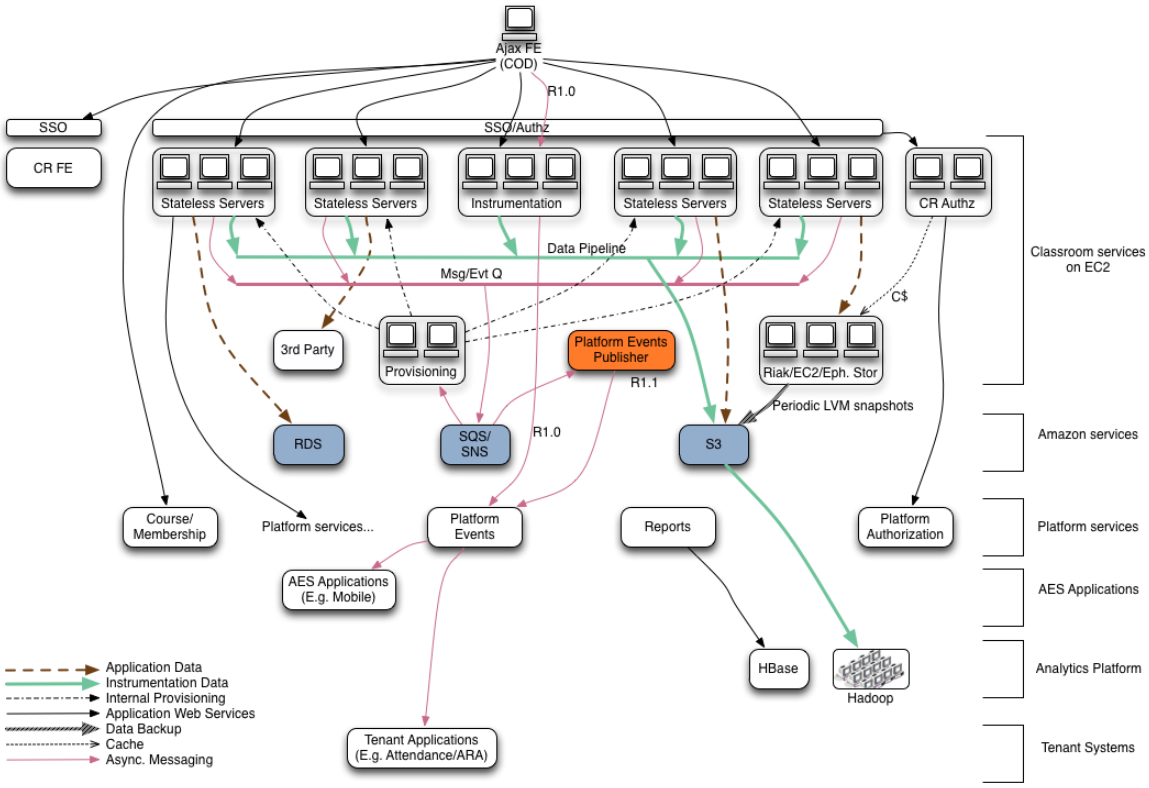


The data produced gets written as Apache Avro logs and eventually gets submitted into Apache Hadoop system. Once the system runs the map-reduce algorithm to crunch the data, the processed data gets persisted in relational store. At this point of time the data is available through Apache Hive.

For a detailed walkthrough of the data pipeline, please refer to Appendix-A.

Service oriented/modularized/componentized architecture

The ALP adapts Service Oriented Architecture (SOA), as mentioned before. Each service operates by contract/APIs. Each service has uniform Representational State Transfer (ReST) APIs defined. The learning platform supports both JavaScript Object Notation (JSON) and eXtended Markup Language (XML) as the content-types. The overall system deployment architecture is as below:



As mentioned earlier, each of the services could be hosted or consumed independently. All of the services follow a cleaner URI pattern that allows easy discovery and consumption.

URI design

```
resource_url ::= "https:" "//" host [":" port] "/" repository "/" version "/"
tenant_name resource_uri
tenant_name ::= "urn:apol:tenant:" tenant_id
resource_uri ::= "/" classroom_urn *resource_uri
classroom_urn ::= "urn:apol:" classroom_scheme ":" scheme_specific_part
```

where

- *repository* identifies the repository hosting the resource,
- *version* is the version of REST service of the repository,
- *tenant_id* identifies the tenant that owns the resource,
- *classroom_scheme* is a repository specific unique scheme name, and
- *scheme_specific_part* is an object identifier unique within the specific hierarchy of the referenced object in its repository.

This enables

- organization of resources into a superordinate-subordinate hierarchy even when the resources reside in different systems

- efficiently implement generic application logic such as authorization at a higher layer without having to access the resources from individual repositories

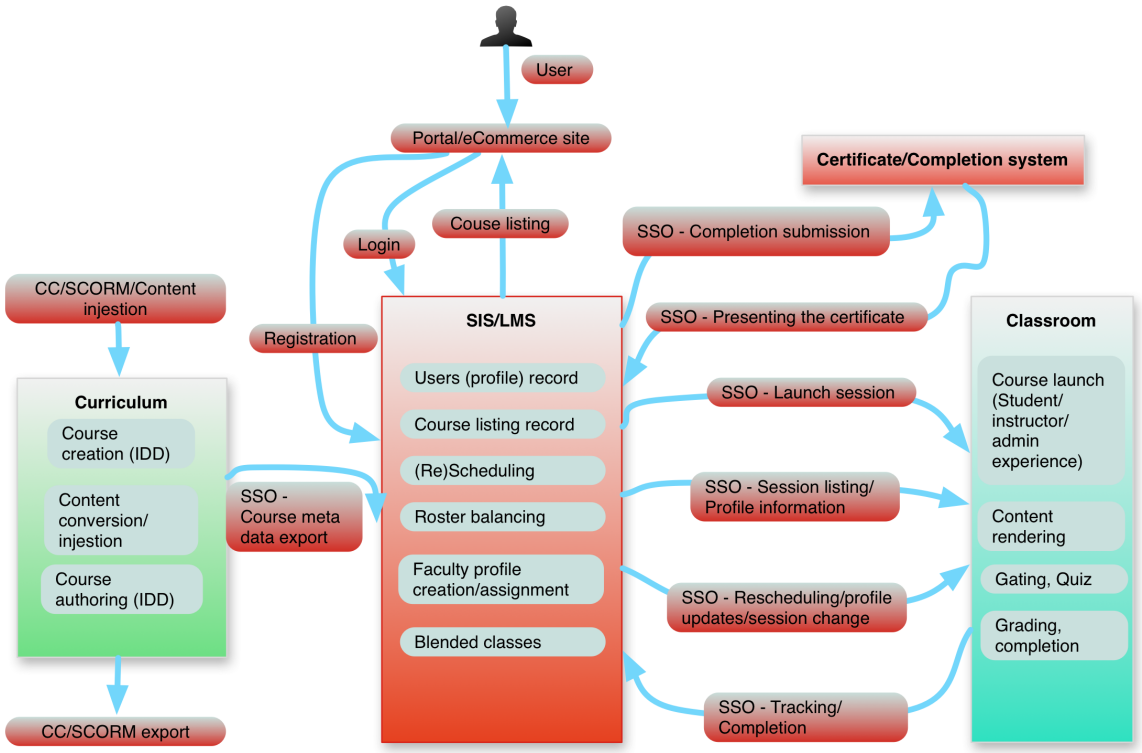
Context

- A generalization of the HTTP Referer (http://tools.ietf.org/html/rfc2616#section-14.36) concept
- Captures application level context from which resources are accessed
- Helps
  - determine the efficacy of course design in achieving learning objectives.
  - identify associations between content, concepts, and learning objectives.
  - deliver courses personalized to the individual needs of students.

1. **x-apollogroup-edu-vlp-client** The value of this header should be the unique id of the client making the API call. This value is optional.
2. **x-apollogroup-edu-vlp-context** The value of this header shall be the object address defined above.
3. **x-apollogroup-edu-vlp-context-ext** The value of this header shall provide additional use case-specific information, and is an agreement between the client and the final consumer of this information. Its format should be defined for the specific use case that requires it, and must conform to HTTP Message Header requirements.

Integration with LMS/SIS

The learning platform has a soft dependency on the learning management system or the student information system. The learning platform accounts for such integration and has clear APIs defined at each touch point. The student/faculty/user onboarding, course selection, course listing, roster balancing and other similar functionalities are expected to be handled outside of the learning platform. A generic overall integration with an existing LMS/SIS system could be shown like this:



All of the integration points define a clear API and have been design with the expectation that they will support integration into the system.

Thin client architecture – application development framework

The learning platform applications are developed using the thin client architecture. The out of the box capabilities of the browsers, are used for developing and rendering the applications, which means that user doesn't have to install any software on the consumer device before being able to access the applications. The preferences set for the user are persisted and applied, which means to some extent the application synchronizes the settings across the devices. The thin client architecture also has the well-known advantages, such as:

- Lower administrative cost.
- Higher security, since the data is centrally managed and the business logic is taken care by the server.
- Local runtime environment stays stateless through the usage of HTTP mechanism.
- Flexibility to behave polymorphic and many more.

The client architecture defines a client framework stack that provides a kick-start and heavy lifting for the application developer. This allows the application developer to focus on the specific application logic. The following are few of the capabilities that the client application development framework provides:

- Instrumentation (BAM – Business Alerting and Monitoring)
- EventBus/Observable objects.
- History manager.

- Permission manager.
- Client side storage.
- BrowserBehaviorManager.
- Transfer request (for AJAX requests).

The client application development framework is built on top of the open source Google Web Toolkit (GWT) framework. Before selecting the framework, a lot of research was put in, lot of Proof Of Concepts using different frameworks (such as jQuery, YUI, home-grown, hybrid) were conducted. No framework could meet every single need that the architecture called for. But we had to pick the one that was in close alignment with the architectural needs, long-term web applications goal and the complexity of the project. The following are few top reasons for making this choice:

- Robust and same development language across that also includes compile time check, static analysis support.
- By providing the basic web development functionalities, it allows the developers to focus on the application logic, instead of trying to find out on how to get there.
- Allows using industry standard Integrated Development Environments (IDE) that allow easy refactoring, auto-complete and similar productivity gains.
- Standard support for unit testing, build tools, obfuscation, minifaction and continuous integration. This is a very important feature, since the regular JavaScript framework based applications would need another way to build, deploy to production.
- Multiple optimization techniques supported such as code-splitting, lazy loading, multiple compiler optimization cycles run during build to find out best possible combinations, remove redundant code to make the runtime fast, browser and language specific files and hence avoiding redundant code being downloaded by the browser, infinite caching possible and other similar features.
- Code generation capability to do the heavy lifting and avoiding writing the same boilerplate code repeatedly. This technique also allows dynamic injection of modules at run time.
- Allows clear separation between application logic and visual representation, so that the right resource can work on the right area.
- If required, external libraries could easily be plugged in using the JSNI technique and similarly the code written in GWT can easily be exported out to native JavaScript.
- Promotes decoupling, modular design and allows enforced API driven contract between developers. In a robust project such as the learning platform applications that contain several hundred thousands of lines of code, it becomes extremely important to keep the code modular so that teams can work in parallel and yet being able to manage the delivery and maintain the same code quality.

- Keep the interaction between the services through Representational State Transfer (ReST) methods. This allows teams to move parallel through a contract driven approach and allows scaling independently.

The learning platform applications are built to make use of the client device capabilities to the maximum and in turn reduce the load on the server for the rendering purposes. This also allows a rich interaction experience to the user through faster navigation, cleaner URL representations, deep linking, bookmarking the application states and other similar Web 2.0 features. For those who are used to the richer applications such as Gmail, Google Docs and similar the learning platform applications look very familiar.

Another important feature of learning platform applications are resiliency to the failures. Unless the bootstrap services such as profile, course offering, syllabus are not accessible, the modules are built to handle the failures in a much-contained fashion locally, without affecting non-dependent modules. With this, the application itself will be usable for the most part even if one of the services/modules runs into technical issues. The error handling is a key feature to the applications and all the modules do abide by the ground rule to provide a consistent experience to the user.

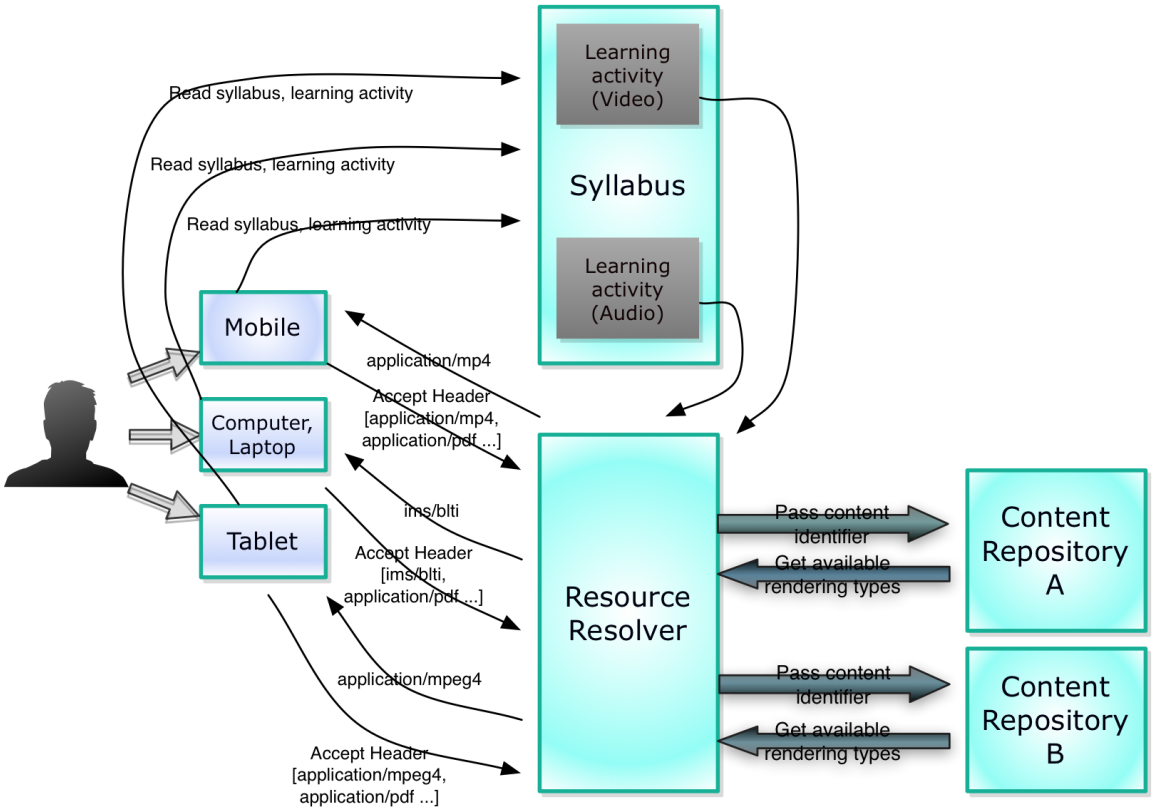
Tools-content extension

One of the core capabilities of the platform is the ability to easily extend on the existing feature. Almost all the components within the platform are designed to extend.

But one of a more common use-case is the required ability for the platform to allow serving new content types. At the core, the platform supports content negotiation by means of having a mapping to different tools against the content type of the content being served. The tools are provided at runtime based on the content resolution.

The platform wraps the content through the usage of Learning Activity that allows additional meta data required for the run time, which could be used for the display purpose or other system integration purpose. The instructional designer would decide what content type has to be used for a particular activity or an assignment and will set the display attributes for the same. The instructor can tweak some of these parameters if instructional designer has set the permission. If there is a new content required to be added, then that would need very minimal code change required, since both the content negotiation (resolver) service and the application support dynamic injection of additional capability to support this.

The content resolution flow at a high level works as below:



The user can consume the course from different device with different capabilities. Learning platform accounts for such expectations through a mechanism known as resource resolver. The resource resolver can be compared to the standard content negotiation mechanism, but with additional capability.

The user loads the syllabus that consists of learning units. The learning units are the logical grouping for learning activities. The learning activities contain the pointer to the resource resolver. The resource resolver has the details about the existence of the content and the source that contains the content.

The application determines the client device capability and invokes the resolver by announcing the capabilities that the device possesses through the usage of Accept header and by passing the different content-types that the client device can accept. The resolver then contacts the content repository in the real time to find out the different content formats available for the content. Now the resolver runs an algorithm to find out the best fit for the device and returns the content-type and the end-point URL where the content could be accessed.

Once the content-type and the URLs are available to the client, the application then determines the appropriate tool that can be used to render the content. The application has a set of tools in the armory that gets chosen as the best fit based on the device capability where the application is being consumed. The tools could dynamically be injected into the application without having to change any

application specific code, through a mechanism known as generators used by the application framework. The following would be an example definition of a 'Tool' that can be used to render a .flv content-type:

```
@Tool(mime = { "video/x-flv" }, sortOrder = 1)
```

The class that has this annotation is representing a tool that can render the video of the type flv. There can be multiple tools available for rendering the same content and hence the sortOrder decides which one should be given priority, if there is a tie in the capability of the device that the user is using. The same tool could actually be used to render multiple content types as well. In other words, the same tool could be used for rendering multiple content-types. Hence, the mime type passed into the tool accepts an array of mime-types as the argument. Integrating a new tool into the learning platform could be that simple!

Due to the dynamic nature of extensibility and the generic behavior while rendering the content, the module that renders the content is known as 'Magic Box'.

The main extensibility point of the Classroom application is its ability to display a multitude of content inline within the Classroom (the so-called "magic box"). The concept would be meaningless if there wasn't an easy and standard way to add new tools to the Classroom's repertoire.

Tools have two distinct sides to them: the viewer used to display the tool in "runtime" mode, and on the other side, the ability to configure the tool. To add a new tool to the Classroom, simply create the required three code files in the source path and recompile the Classroom. The new code will automatically be picked up and added for use.

Viewers

A tool viewer is the main component of any tool. It is what actually provides the user interface for the tool so users can interact with it. A tool might be as simple as a snippet of HTML providing a link to an external site; a frame that displays another site, an embedded Flash player, or the tool could be as complicated as a full-blown application, such as a quiz. In general, viewers can be considered applications in their own right.

The viewer to use is selected based on the Content-Type for the learning activity or assignment. The Content-Type maps to the mime-type for individual files, such as PDF documents and Flash SWF files, but is often a custom type for tools.

Configuration Information

Configuration information is used to display basic information about the tool, what it does, and often provide either a static or dynamic preview in a selection dialog. Often, configuration information is simply a localized chunk of HTML, which describes what the tool does, wrapped with the appropriate interfaces information.

Configuration Options
The configuration options allow the user to specify configuration options and parameters for the tool when the tool is added to the syllabus. Not every tool will have a visible user interface that the user can configure, or certain configuration options might be disabled or hidden, based on the user's permissions.
For example, adding a textbook to a course requires a pointer to the book, but the user might not always be given a UI element to change the pointer. Often, a search interface is provided instead. The search functionality then sets the underlying structure directly and changing the object (the book in this case) might require the user to remove the existing item and perform a new search, essentially adding a new instance of the tool.

ActivityInfo
The ActivityInfo data defines the parameters that are required to display a tool and its content. It is an extensible key/value pair dictionary that allows tools to have custom parameters. In addition, certain keys are reserved for use by the host (Classroom and CourseBuilder) to change the behavior of the tool.

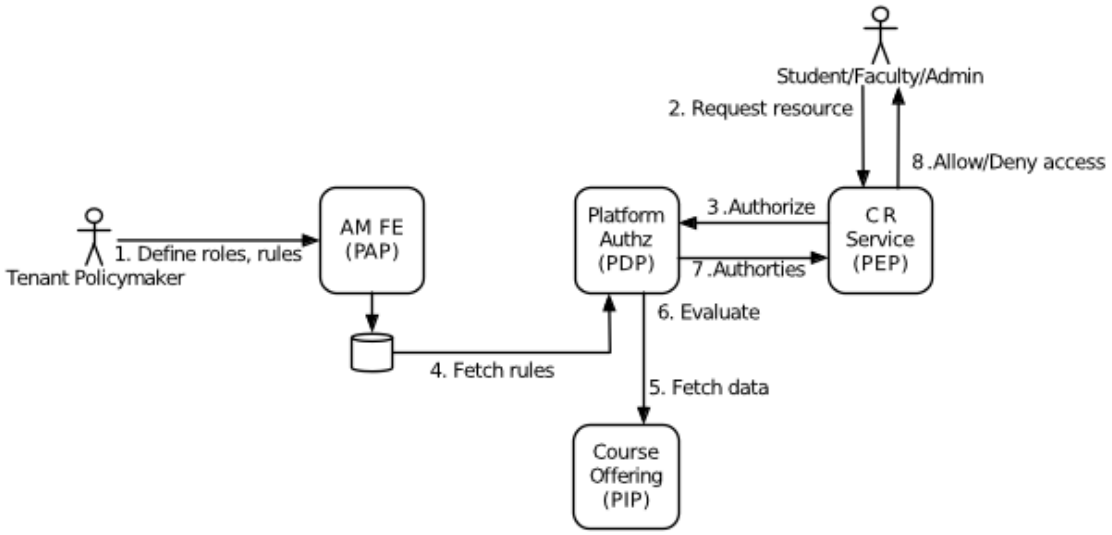
Pre-Defined ActivityInfo Parameters
The following table lists the predefined ToolInfo parameters. Tools should clearly document tool-specific parameters in their documentation.

Key	Description
AllowComments	When set to true, displays the comments module beneath the tool.
AllowDownload	When set to true, a link to download the content for the tool is displayed.
CanResize	When set to true, allows the user to resize the height of the tool.
CanExpand	When set to true, displays the Expand option that allows the user to maximize the tool to fill the display.
Width	An HTML width value to set for the content. This option only has meaning for certain types of tools, such as Flash embeds where the width of the content can be specified. This option does not affect the size of the tool container, which is determined by the layout and whether the PeTs dock is expanded.

Height	The initial or fixed height of the tool. The height will be fixed when used in conjunction with the allowResize option set to false.
FriendlyName	The short, user-friendly name that describes the tool or content, such as eBook, Video, or Quiz.
ToolStyle	The CSS class name used to determine which icon to display for the tool.

Permissions/Authorization driven

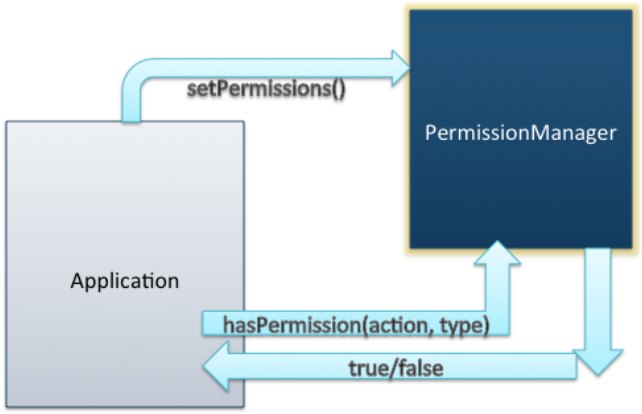
The ALP is controlled through permissions at a very granular level by a component known as Access Manager (AM). The platform has the notion of being polymorphic from the beginning. Essentially, what this means is the services/applications are organized in such a way that even though the same entry points, URLs or modules being accessed, they provide completely customized experience, based on the logged in user and applicable context, at that point of time. The authorization framework implementation is comparable to the industry standard eXtensible Access Control Markup Language (XACML), makes use of Drools Expert Engine and Drools Govner. The general flow is defined by:



The Access Manager (AM) itself is decoupled from the ALP for the very purpose that the permissions could be managed or controlled independent of the core platform components by the authorized personal, such as a tenant administrator.

The service APIs makes use of the Spring framework security annotation to handle the authorization. The permissions govern the usage for the current context based on the access and role applicable.

On the client side, the Access Manager component is complimented by the Permission Manager component. The permission manager gets fed with the applicable set of permissions at the bootstrap time and subsequently governs the access to the components through API driven mechanism. The application also eliminates serving the not relevant or not permitted component/feature to the browser, by a mechanism called code splitting. The application implementation of the Permission Manager could be depicted as below:



Configuration

All the services and applications follow configuration driven approach for supporting additional use-cases. The configuration controls the runtime behavior for a particular use-case, context, user etc. The configuration is maintained outside of the actual service or the application itself, such that the configurations could be tweaked where necessary. The end points for communications are derived off the configuration as well. This means that if the end point has to updated to support a runtime change, then this could simply be handled outside of the application or service code and the functionality would continue to work as is, as if nothing changed.

Customization-theming

The applications follow the standard way of development by separating the static assets outside of the code. This would allow the customization or theming a breeze. If required a User Experience designer could revamp the entire User Interface (UI) with a new theming or customization, without having to change any code.

Internationalization/Localization

The applications support internationalization and localization by virtue of the development framework capability. The messages are externalized and allow easy updates. The layout organization also accounts for different localization expectation. The entire application supports standardized display of institution time or campus time where the course is being taught. Throughout the platform the dates and time

are being persisted in ISO 8601 standard. Only during display the appropriate zone information is applied and rendered to the user.

Security

The learning platform has stricter security guidelines against the URL declaration, data access, cross-site scripting and other security areas. Each of the critical resource is being protected behind the authentication filter, so that they are available only for the logged in user. The resources are also accessible for the personnel with appropriate authority. The security filter within the container adheres to Open Web Application Security Project (OWASP) standards and provides the security measures against the malicious activities. The application framework has built-in ability to handle cross-site injection in the content by virtue of Safe-Templates.

Accessibility

The learning platform applications undergo stricter accessibility audition from a dedicated team of engineers to maintain Section 508 compatibility. Although, the applications haven't yet gone through the certification yet, majority of the modules and the applications in its entirety are adhering to Section 508. The modules and applications are accessible through keyboard and other accessibility compatible software, such as JAWS. The accessibility is provided ground up using the client application development framework and also the controls that the team creates, incorporate the accessibility built into the components so that they can be re-used.

Multi-tenancy

All the services and applications in learning platform have multi-tenancy incorporated from the beginning. With multi-tenancy, the repositories follow a cleaner data sharding, the services provide complete isolation between each tenant's data and the applications allow clear customization for each tenant.

The tenantId is used as the trigger for serving different facets of the application to the user. Each user has an association with the tenantId that decides the execution path for the user. The hosted application internally has the clear separation defined to serve the tenant specific customization to the user.

Code standard and unit testing

The services and applications within learning platform follow well-defined coding standard. The developers make use of the standardized plug-ins for enforcing the same standard across.

The builds running during the continuous integration process catch the violation and alert the respective teams, so that the teams will adhere back to the process. The plugins such as FindBugs, CheckStyle, and Emma etc. are used for serving this purpose.

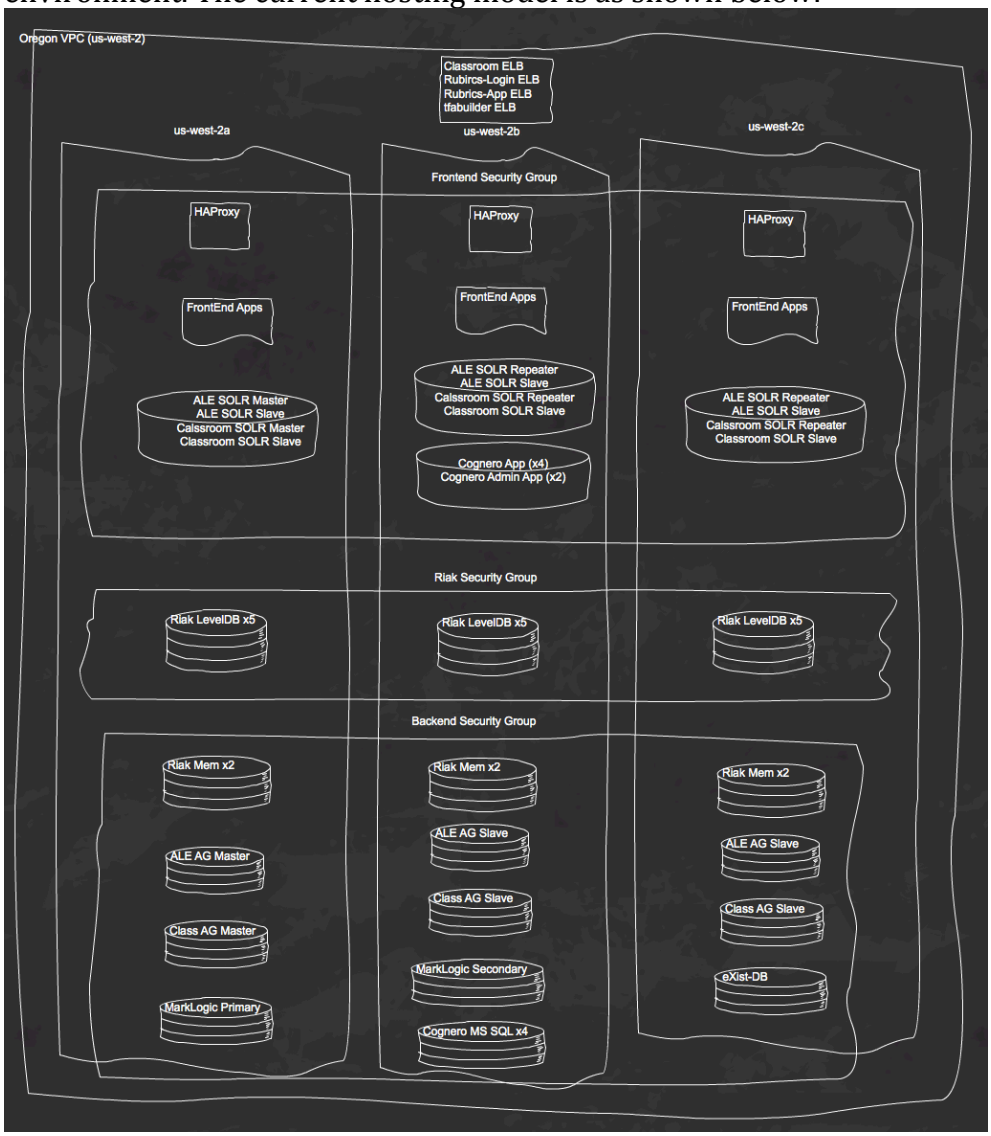
The amount unit testing is analyzed against the code coverage, which is set at a good > 90% range. The teams make use of JUnit, TestNG, HttpClient, GWT framework, Selenium, LoadRunner and other tools for various testing needs.

Hosting model

The learning platform applications are hosted in the following order:

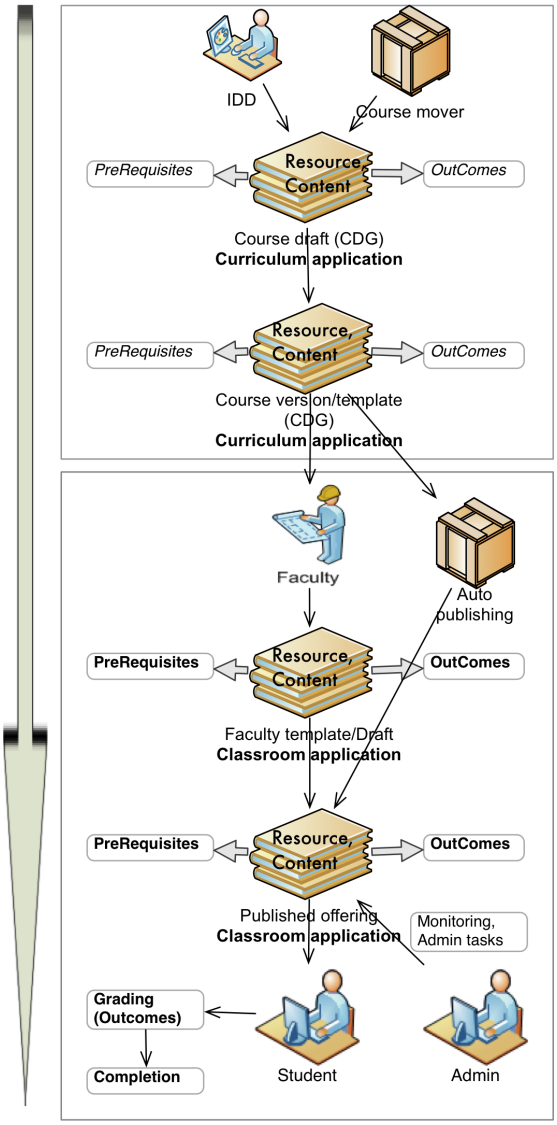
- Development environment.
- QA environment.
- Staging environment.
- Production environment.

All these environments simulate the same setup as the production with the exception that the number of instances varies in the development and QA environment. The current hosting model is as shown below:

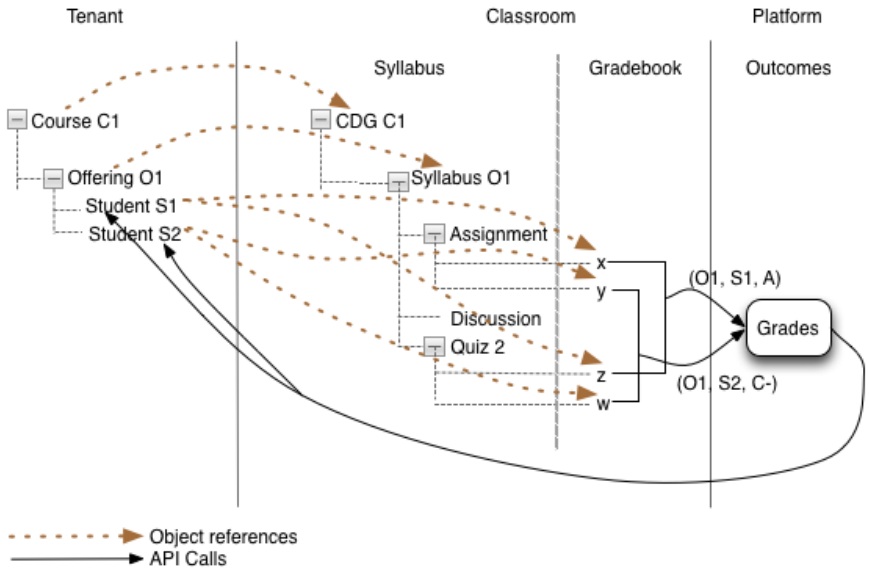


Overall system flow

The learning platform revolves around Syllabus structure. The Syllabus traverses through different lifecycle during the course and also holds different systems together. The following diagram shows a simplified version of syllabus lifecycle:



Another way to represent this including the tenant capabilities would be:



The course originates as a Course Design Guide (CDG) from the instructional designer. The instructional designer would work through the CDG draft version over a period of time, before publishing a version of the CDG and making it available to the instructor. If the course existed from another source, the course mover tool moves it over to the Curriculum structure. Once it is moved, the instructional designer can then work through the course and then publish a version to the downstream systems. This is the step where the instructional designer sets the pre-requisites that the learner should have before taking the course and also the outcomes that the learner would gain out of this course.

The instructor can then create a profile template to adjust the syllabus where the policy allows or relevant and preview the template. At this stage the scheduling system hasn't yet assigned a course offering and assigned the instructor into it. But the instructor knows ahead of time that he is going to be teaching that particular course in the future. Each template is individualized so that different instructor could have his version created, but the same instructor can also create multiple templates in the process. Instructor is not making change to the pre-requisite or the outcome of the overall course structure.

Until the offering is published, if the students login to the class, based on the permission window configured, the students will see the CDG version of the syllabus.

Once the scheduling system creates an offering, the instructor can then create a profile draft and preview, before publishing it and make it available to the students. The editing could be done once after the offering is published, if the policy allows. E.g. For University of Phoenix, the instructor editing is allowed through policy, after the course is published for the future weeks.

The auto publishing kicks in case the instructional designer doesn't get assigned by the time the offering begins. The auto publishing would create an offering from the current effective CDG version and make it available to the student.

The syllabus consists of learning activities that are essentially the wrapper around the content. There are different types of learning activities: ToolActivity, TaskActivity, ResourceActivity, AssessmentActivity. Each type of activity is geared to serve a specific purpose. The ToolActivity is the learning activity that contains the information regarding learning tools that can be used to consume an activity. The TaskActivity is the learning activity that is created when either an instructor or student has a task that needs to be accomplished as part of the syllabus. The ResourceActivity contains resources for either instructor or student. The AssessmentActivity is the learning activity that yields points to the students, by virtue of providing them with the assignment. The activities can be bifurcated into required and recommended activities. The required activities are a must complete for meeting the objective in the course by the learner, while the recommended activities are additional and good to complete. The assessment types of activities support both individual and team kind. The teams should complete the team assignments, working together.

The syllabus consists of learning units that in turn contain the learning activities. The learning units are a collection of learning activities that could either be compared to lessons, weeks, days etc. The learning activities contain information for the applications to both display the content and point it to the resource resolver service that helps negotiate the content. The content is then displayed in a uniform way to the end user in the Tool Container module (or popularly known as Magic Box).

Once the student completes an assignment the system allows integration from various assignment systems into grading tool. The instructor gets to use the Rubric system as a mapping for the grading and provide the assessment.

Please refer to the Appendix-B for the detailed walk through of individual service dependency diagram, which also highlights overall workflow.

Integrating external content into learning platform

The Classroom-Curriculum applications enable integrating external contents easily, in a consistent fashion and yet providing unified experience to the end-user, since this is a common use-case for any such platform.

There are multiple ways to integrate and external content:

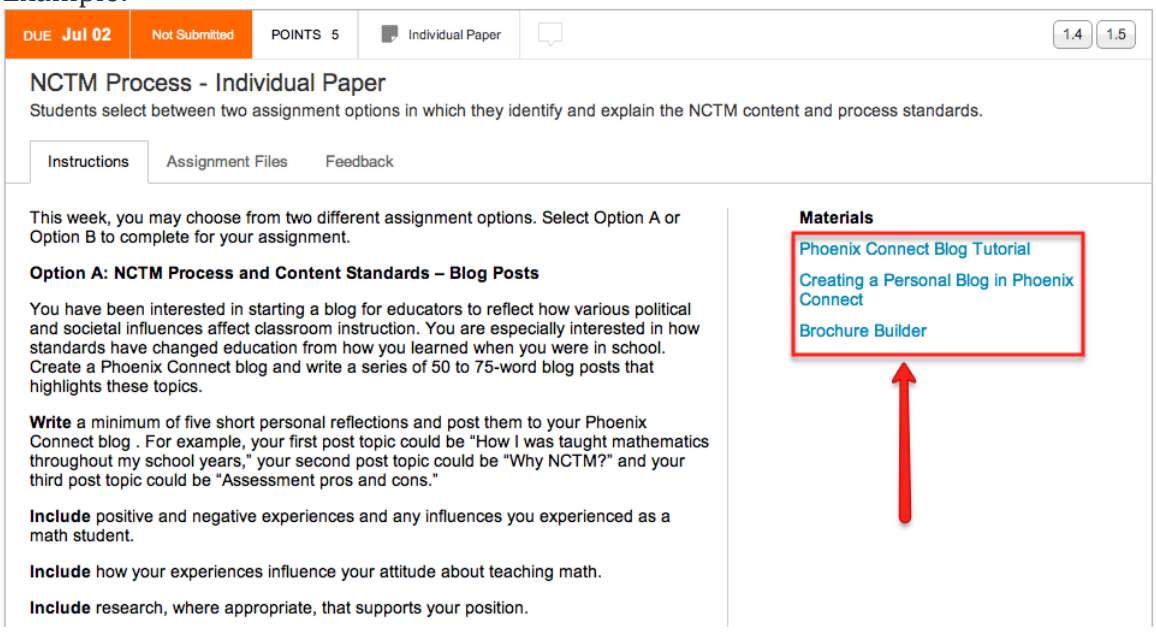
Pointing to External Content

This is the simplest of the external integrations. The content could be presented as a supporting material or a resource to the user. In this case, Classroom application will simply have an anchor (link) pointing to the external content that will open up in a new window.

In this case, there won't be any interaction between learning platform and the external content/provider. Learning platform will not take care of any SSO (Single Sign-On) integrations between Classroom and external content.

Content displayed in this manner (the link text and URL) must be included in the Syllabus data (generally in 'extensionInfo' object).

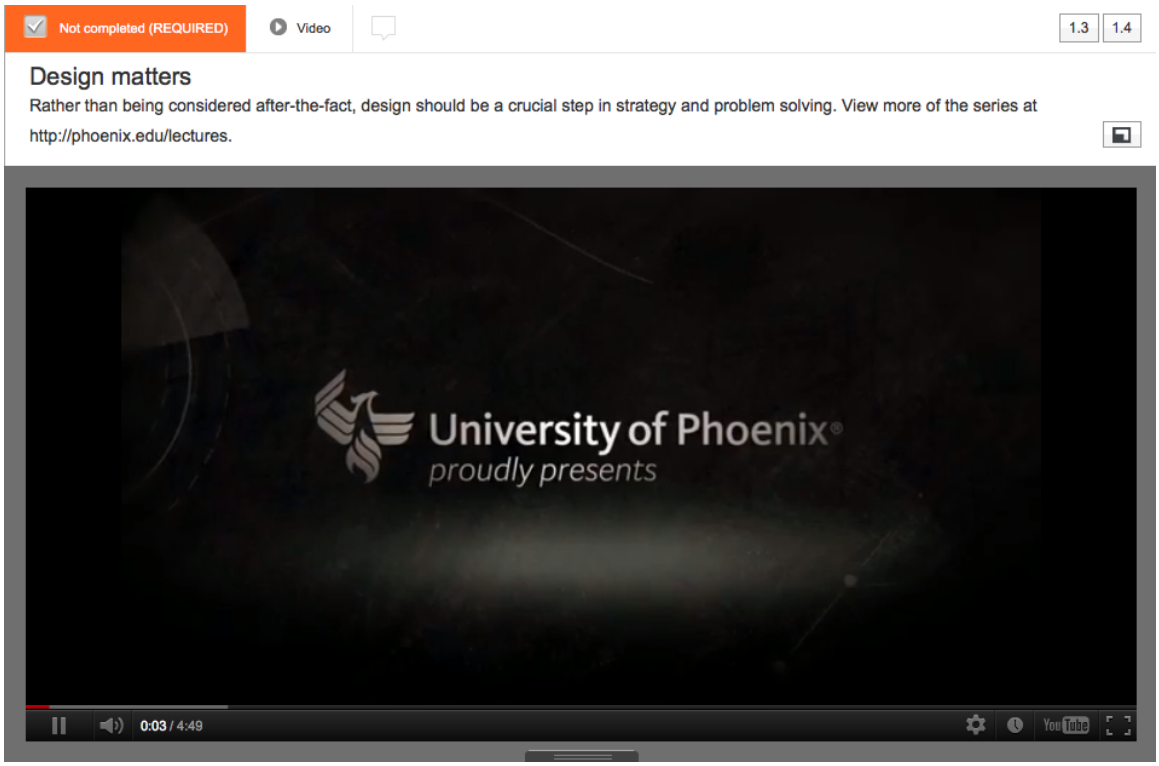
Example:



Displaying Content Using an IFrame

When the external content doesn't need any input from learning platform during launch or doesn't need to interact with learning platform once it is done, but can operate on its own, then we can load the content inside the magic box within the IFrame. This will be provisioned inside the syllabus data as an 'IFrameActivity'.

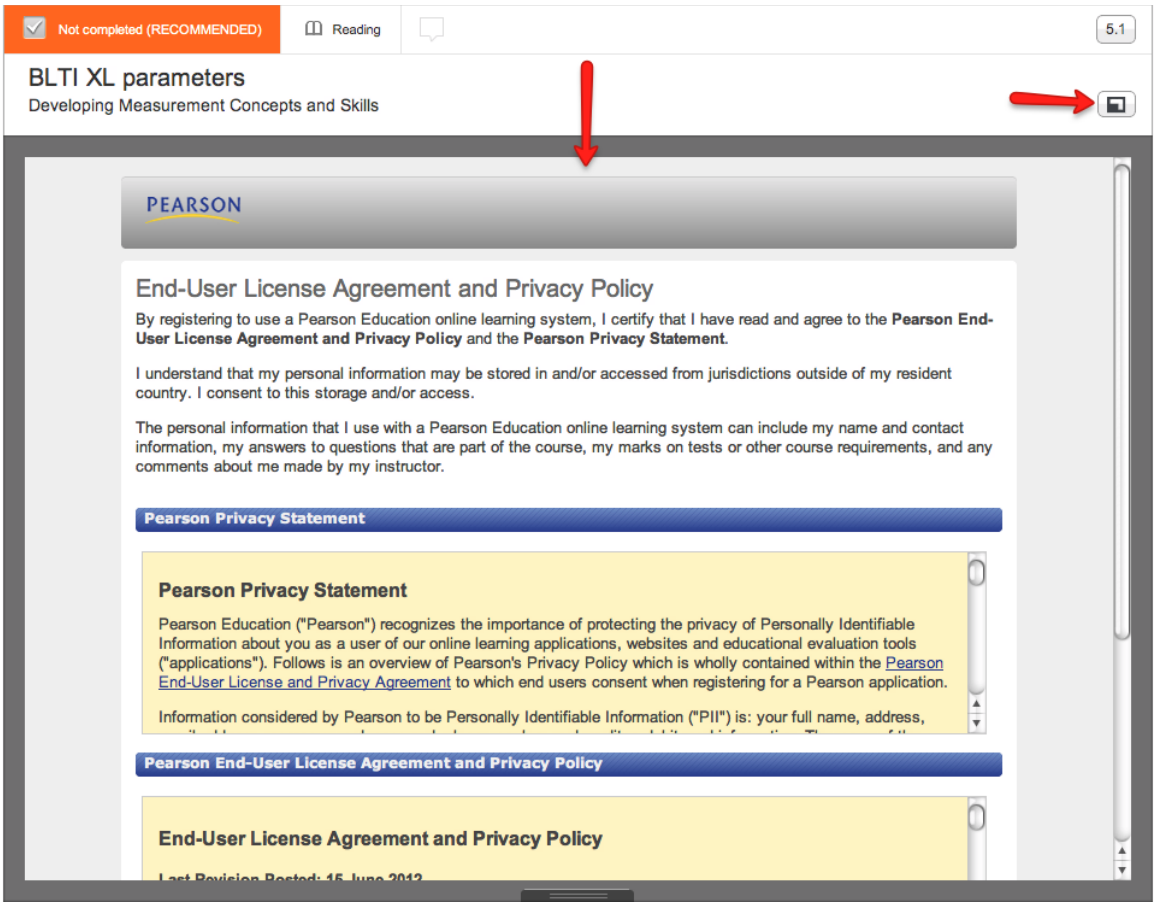
Example:



Basic Learning Tools Integration (BLTI)

This is the case when the external provider already has a tool that renders the content that learning platform needs to consume. Learning platform recommends usage of IMS standard BLTI technique to integrate between an external provider in this case, since it has options available to pass any data that the launch requires and callback to end the interaction in a standard defined way. The launch also includes certain mandatory fields as required by the BLTI implementation. This will be provisioned inside the syllabus as a BLTI activity.

Example:



BLTI Wrapper

If the external tool doesn't provide a BLTI implementation the learning platform would provide a wrapper BLTI provider project implementation that can internally invoke the external content with different ways of maintaining the SSO. This includes enough data for the tool to consume the data needed to interact and then the tool provider would need to maintain the SSO session internally and keep it transparent.

The advantage of using a BLTI wrapper around the external content is downstream learning platform systems doesn't have to establish a different way of dealing with a new external system. Curriculum application would configure the tool as a standard BLTI configuration, Classroom would launch it as a standard BLTI interface, Resolver would deal with the type as a standard BLTI launches. The integration with the external tool would only be applicable at the wrapper tool level.

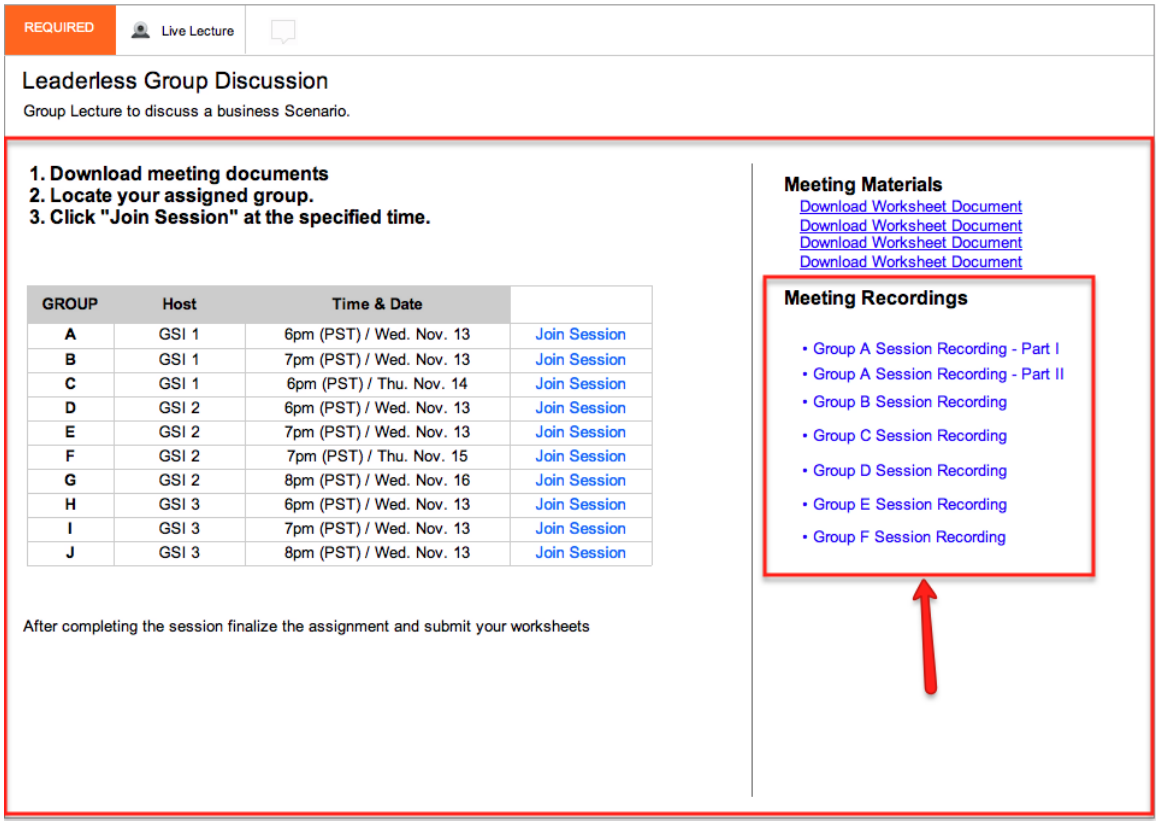
Custom Tool for External Content or 3rd Party Application

This is mostly the use-case when a third party content provider is well integrated into Classroom infrastructure that exposes APIs to consume/produce data, mostly

through a proxy or other mechanism. In this case, the assumption is that the proxy would take care of the SSO and other integration issues and expose well-defined REST based APIs for the Classroom application to invoke. Essentially, since a proxy is setup in the learning platform environment in front of the 3rd-party content, from the learning platform perspective it is interacting with an internal service. This means that although behind the scene the provider is external, the data/content can be consumed in a format that learning platform can understand or deal with.

An example of this type of integration is our custom user interface for the Cognero quizzing application. However, integrations can be much simpler such as displaying a simple set of data retrieved from the proxy and displayed with data stored within the Syllabus.

Example:

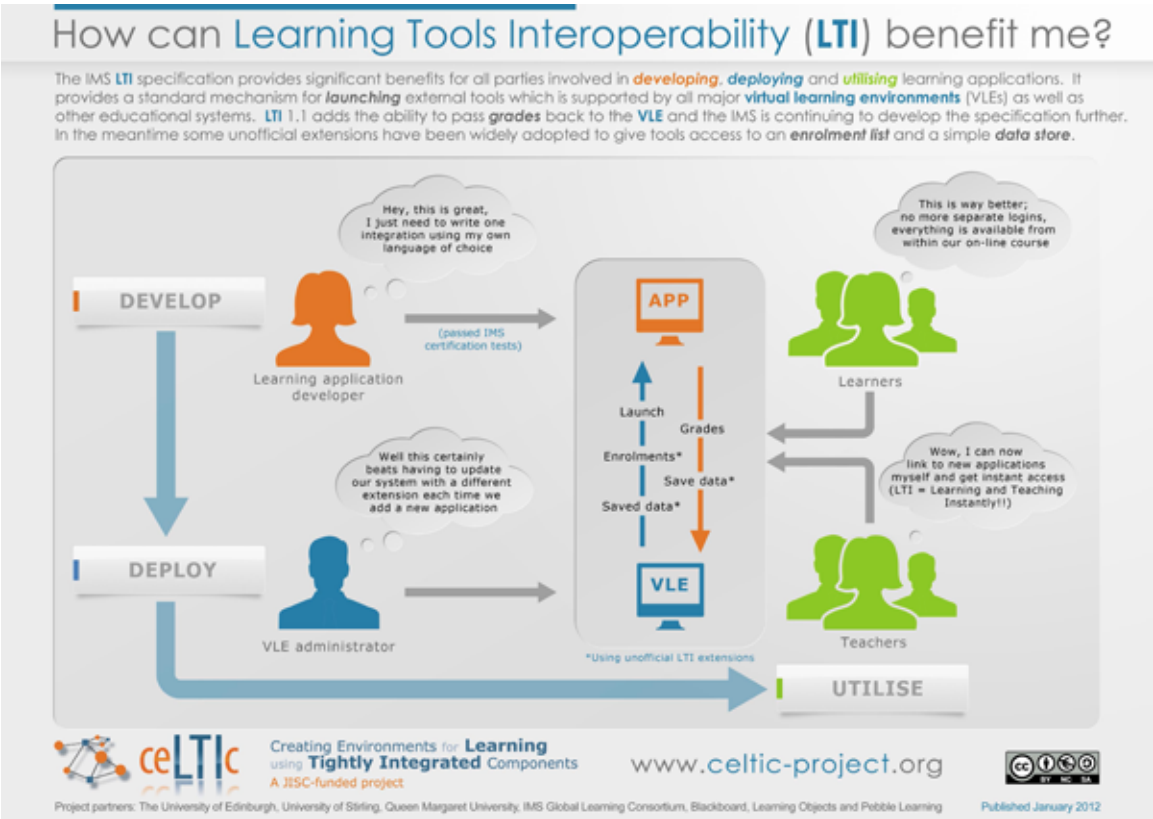


IMS Global Learning Consortium Compliance

The learning platform is compatible with the IMS consortium's Learning Tools Interoperability (LTI) standards.

The learning model is loosely defined on the LMS's Course Management Service Information model.

The following diagram depicts the overall flow of the LTI tools:



The content standards are based off Doublin Core (DC) schema.

Learner profile

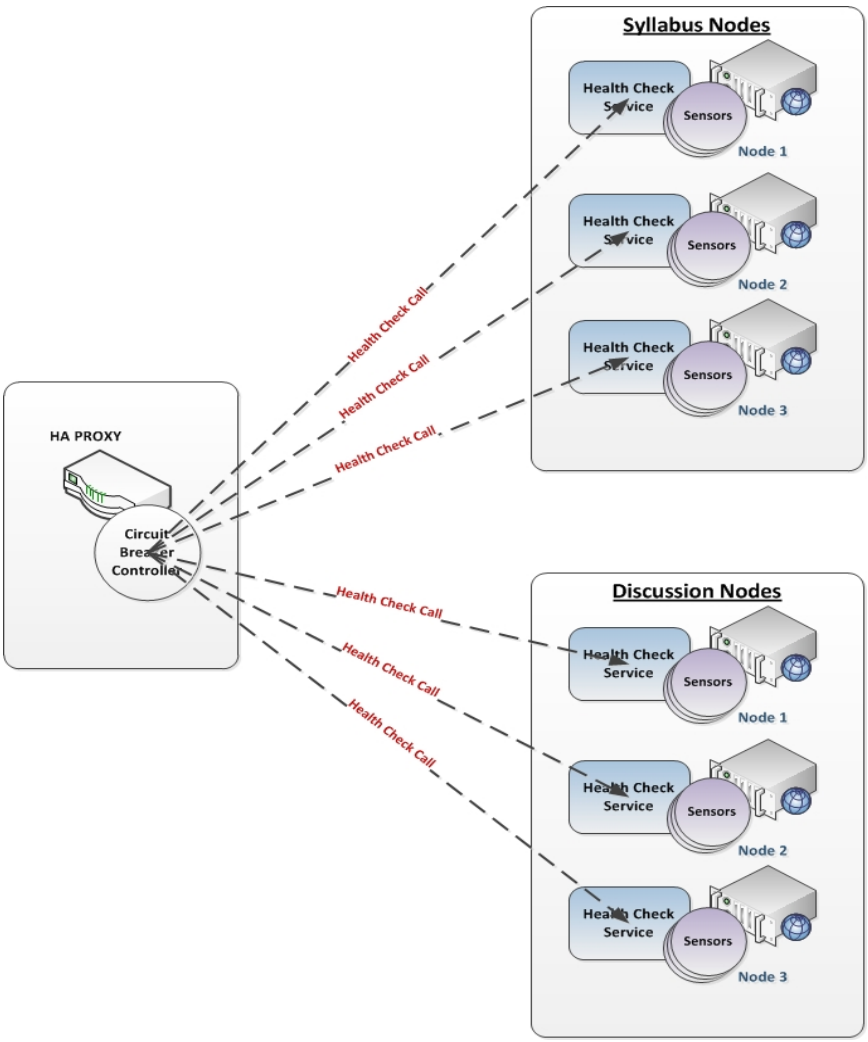
Platform components overview

Infrastructure overview

- Container.
    - The learning platform container is built on top of the GlassFish open-source application server, provided by Oracle Corporation. GlassFish container makes use of the Apache Tomcat derivative as the servlet container for serving web content with an added component called Grizzly which uses Java NIO for scalability and speed. It also supports Apache Felix OSGi run time.

      The base container has been customized to include the components that provide majority of the basic functionalities that are needed for the services/applications.

- Authentication:
    - The learning platform provides the authentication (SSO – Single Sign On) client out of the box. The SSO client is dynamically instantiated, which means it could easily be provided on top of any other system where it needs to integrate into.
- Authorization:
    - The learning platform container provides authorization part of the container. The authorization client is also geared to be easily instantiated with different implementation where required.
- Instrumentation:
    - One of the key components, known, as Instrumentation is part of the learning platform container. The services and applications can easily make use of this component. All of such dependencies are available through dynamic injection and dependency management framework (e.g. Maven).
- Circuit breaker.
    - Circuit breaker allows load balancers to get health status of a node and thus can control traffic to the node. The circuit breaker also provide option to block service calls to nodes when they are unhealthy. The following diagram shows the working of circuit breaker:



This uses a combination of a health checker service and load balancer to achieve the over all circuit breaker pattern. Mainly there are 3 components in the solution:

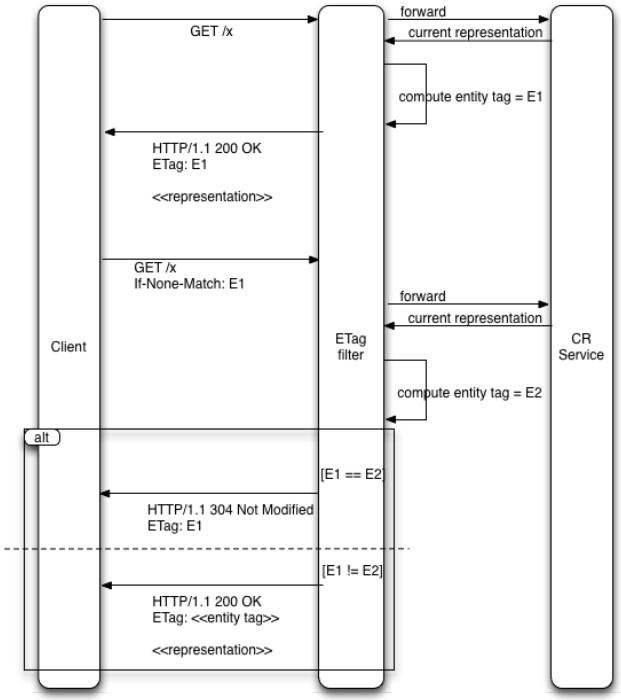
Circuit Breaker Controller: The controller is built/configured on top of HA proxy and has the responsibility of closing and opening the circuit.

Health Check Service: Health Check Service provides health information of a particular node. Circuit Breaker Controller will call this service to get the health information. Health Check service uses Sensors to get specific health information.

Health Check Sensors: Sensors monitor and provide health information of specific area.

- Alerting:

    - Messaging queue is implemented using Amazon Simple Queue service(SQS) and Publish/subscribe messaging model is implemented using Amazon SNS.
- Notification:
    - Amazon SNS uses push mechanism to send the message to its list of subscribers. Classroom client registers with messaging API as a subscriber to the topic to receive the message and message handler for processing of message. On receipt of message to the topic, amazon SNS delivers message to the messaging endpoint, using http post. Messaging implementation invokes appropriate messaging handler to process the message.
- Distributed cache:
    - Learning platform supports distributed caching and uses Riak in the background. It is compatible with JSR107. It is implemented in a way that the underlying storage could be switched without affecting the consumers.
- ETag solution for utilizing the caching:
    - The HttpCacheService OSGi Service bundle contains the ETag Service, is implemented as a filter. Any service that needs to send an ETag in its Http response should add an ETag filter in its web.xml. When a request comes into that service, and the service builds the response and sends it, the response passes via the ETag filter. This filter computes a hash using MD5 and adds it to the http header of the response. When a request comes in with an ETag value (the http header parameter is If-none-match) the service once again builds the response data. This response data then goes to the ETag filter which computes the MD5. If the computed MD5 matches the ETag values sent in the request, it means the data has not changed. The ETag filter sends an http status of 304 instead of 200 and the response data is not sent. Thus saving on the transport time for the data.



- Security filter:
    - Web applications require the data to be sanitized and filtered out before they can be stored or retrieved to keep the data safe. The learning platform provides integrated support for Apollo Reference Monitor for Secutiry (ARMS) filter. The filter helps to protect from Cross-Site Scripting (XSS), request forgery, SQL injection and other similar security threats.

      This handles:
        - URLs.
        - Input data.
        - Request headers*.
        - Request body*.

      The security filter makes use of 'x-requested-with' extended http header to overcome the cross-site request forgery. The applications that make use of the services pass this header with a value that the applications have received from the security token. The filter than verifies the authenticity of the token by the same value to make sure the request hasn't been forged.

- Data store drivers:
    - Since Riak is used by several services, the Riak PB Client driver is shipped as part of the learning platform container. Riak is a dynamo style, distributed key-value store that provides a map reduce query interface. It exposes both a REST and protocol buffers API. This is a Java client for talking to Riak via a single interface, regardless of

underlying transport. This library also attempts to simplify some of the realities of dealing with a fault-tolerant, eventually consistent database by providing strategies for:

  - Conflict resolution
  - Retrying requests
  - Value mutation

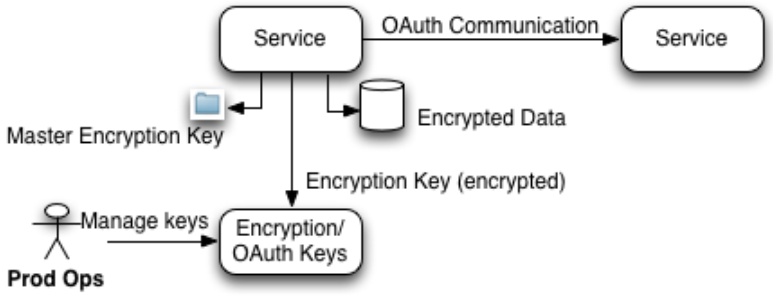
- Logging:
  - The learning platform container supports logging through Logback. Logback's architecture is sufficiently generic so as to apply under different circumstances. At present time, logback is divided into three modules, logback-core, logback-classic and logback-access.

    The logback-core module lays the groundwork for the other two modules. The logback-classic module can be assimilated to a significantly improved version of log4j. Moreover, logback-classic natively implements the SLF4J API so that you can readily switch back and forth between logback and other logging frameworks such as log4j or java.util.logging (JUL).

- Application encryption/OAuth:
  - Application Encryption/Signing is used in two contexts:

    - OAuth comunication between Classroom components.
    - Encrypting application data at rest: This should be provided at the System level by encrypting data at the Storage layer. But our technology choices (Riak, Mongodb, RDS), and company-wide System services do not provide this capability.

    The following figure shows Key Management and Protection in Classroom:



- Encryption and OAuth keys are stored in a central repository.
  - These keys are encrypted using a Master Encryption Key.
  - Access to the central store is restricted to only Classroom Service nodes.

- Master Encryption Key is deployed to every Service node and stored in the filesystem of the Service node.
    - Access to this file is restricted to only the system users that applications run as. For web services, this is the glassfish user.
- An application that needs to encrypt/sign data, either for storage or for communication
    - Fetches the appropriate Encryption/Signing Key from the central store,
    - Decrypts the key using the Master Encryption Key,
    - Encrypts/signs the data using the Encryption key.

Services overview

Syllabus repository/service

Introduction

Syllabus is the heart of the learning platform. It connects different applications together; it connects different systems together and drives the holistic learning experience for the users.

What does it do?

The service is needed that in order for the system to be able to support different workflows, connect different systems together and support different users, we need a data structure that is cleanly modeled and drives the contract between systems, yet allows extension and is flexible to accommodate new content to be included as part of the syllabus.

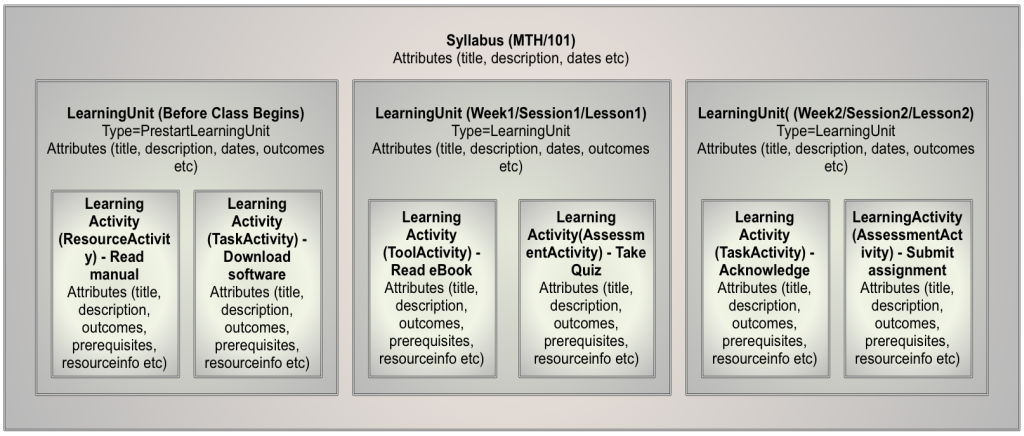
The syllabus is the central contract between learning platform services and the applications, entry point for all the other systems, such as discussions, assignment upload, assessment, contents, resources etc.

The syllabus supports the full life cycle of the course object.

Architecture

Syllabus contains all the elements that you would imagine in a regular syllabus. It contains the meta-data attributes for the course being taught (title, description etc). It contains LearningUnits that allow logical grouping of LearningActivities. LearningActivities are the wrapper around the content, tools, assignments that a learner would need to take.

The simplified syllabus model looks as below:



The syllabus model consists of the Syllabus container that has several attributes such as title, description, course code, dates, extensionInfo etc at the top level. The syllabus object contains one or more learning units. The learning units are the logical grouping for the learning activities. The learning units contain attributes such as title, description, outcomes, prerequisites, extensionInfo etc. The LearningActivities in the Learning Unit are a pointer to the learning activities in the Learning Activity Repository (LAR). The Learning Activity includes title, description, outcomes, prerequisites, activityInfo, resourceIno, extensionInfo etc. As can be seen the model includes quite a few of flexible, extensible structure in the form of name-value pair definition.

The outcomes are a crucial part of the model. The overall outcome for a course is defined as an aggregation of the outcome that a learner meets at each learning unit level. The outcomes are defined at a learning unit level that point to the meta-data repository where the ontologies for the outcomes are defined. The learning activities also reference the outcomes that are defined in the learning units. In other words, the outcomes in the learning units gets associated with the learning activities in the learning units, although not all activities are mandated to have an outcome.

Syllabus also supports several workflows to handle the lifecycle of a course. Please refer to the overall system flow section for the detailed information.

For a detailed syllabus and learning activity model, please refer to Appendix-C.

Learning Activity repository/service.

Introduction

Learning activity repository is the central repository for all the learning activities. The activity objects are maintained independently from the syllabus to allow consumption of the activities outside of the syllabus.

What does it do?

Technically, external vendors or content authors could create the activities, well before a syllabus is created. The syllabus can then make use of these activities through search of meta-data and other similar mechanism. Theoretically, it allows content exchange capability as well. But the surrounding infrastructure hasn't been established yet.

Architecture

The Learning Activity contains mainly of the following types:

- TaskActivity: The tasks that an instructor or a student needs to complete. E.g.: Download software.
- ResourceActivity: The type of activity is mainly intended to be used to provide materials. E.g.: Familiarize yourself with the textbook used in this course.
- ToolActivity: The type of the activity that is used to provide information to the applications on usage of the tool. E.g. eBook chapter-1
- AssessmentActivity: The type of the activity that is used for providing assessment capability and generally includes points and due dates against the assignment. E.g. Quiz, Assignment upload etc.

The activities could either be categorized into required or recommended based on the flag: 'required'.

The following are the examples of content types that are currently supported through learning activity (although extending this is simple):

- Ebook (such as pdf, xml-ebook etc)
- Video (such as mp4, mpeg4, flv etc)
- Audio (such as mp3 etc)
- External links (such as a website etc)
- Embedding (such as Youtube etc)
- Synchronous learning (such as Adobe)
- LTI integration (such as Pearson, Wiley etc)
- Quiz
- Assignment upload
- Discussion questions
- File uploading type

Resource resolver service

Introduction

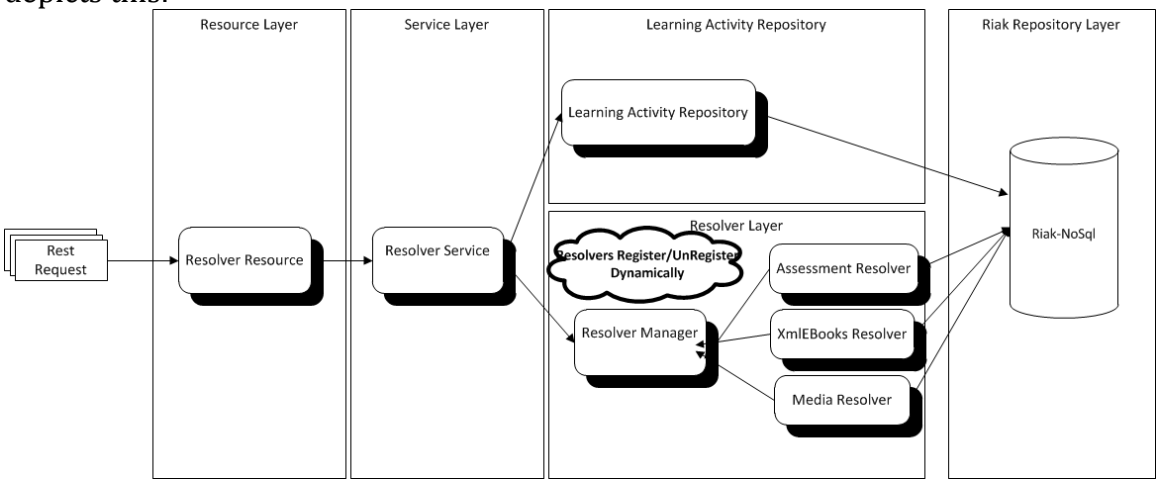
The resource resolver is the content negotiation service for the learning platform applications.

What does it do?

Content negotiation is an important part of the content consumption life cycle. Some of the combinations that a user would consume the applications are different devices, different behavior, different environments and different user-agents. With so many variations in the consumer device, we need a standardized way for providing the consumption end point and the type of the content that is applicable based on the run time. Please refer to the tool-content extension section for the details.

Architecture

The resolver service needs to be flexible and dynamic and hence it makes use of the dynamic capabilities that the OSGi container provides. The following architecture depicts this:



Content provider service

Introduction

The content provider service provides the ability to configure different providers for each content type. This is also used for providing the LTI configurations.

What does it do?

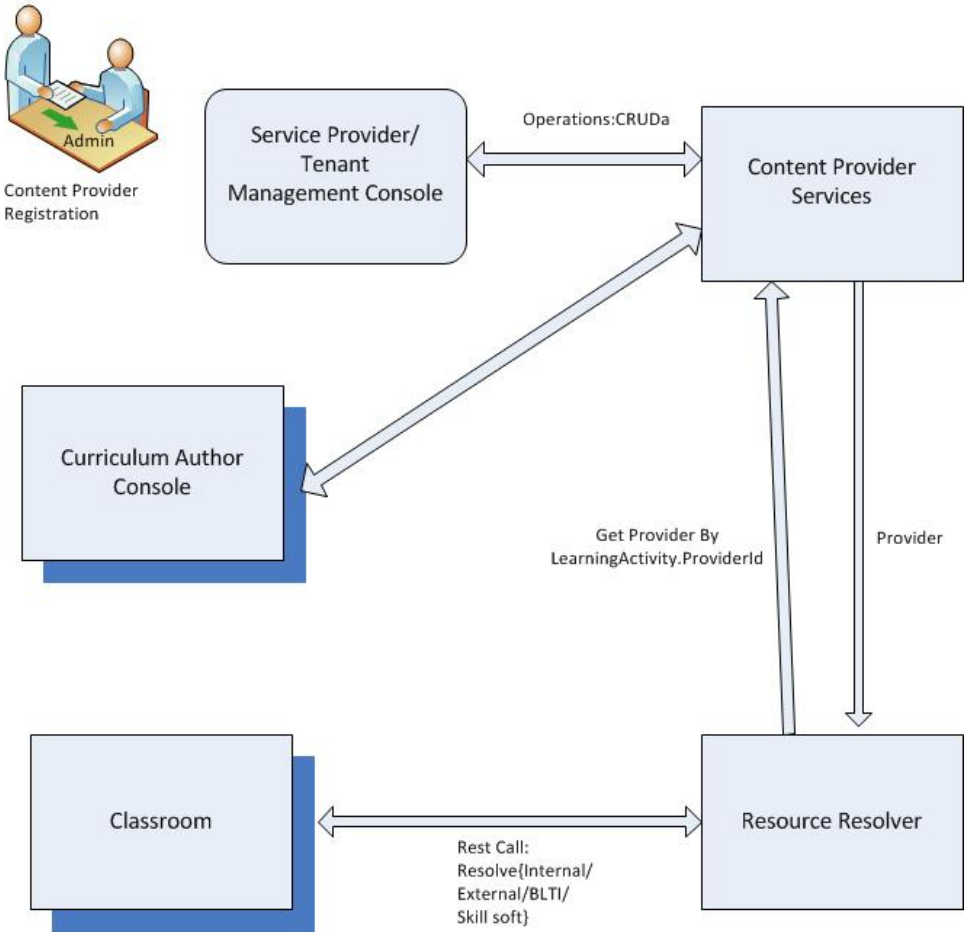
There can be multiple providers for different content types. E.g. A video content could be provided by the internal Content Pipeline or an external vendor. We will need to configure the ways to consume the content from the provider. These configurations are managed using the content provider APIs.

Architecture

The tenant administrator would be having the contract with the provider to provide access to a content type. During this process, the provider would provide details such as the URL to be invoked, parameters to pass and where applicable certain authentication or authorization parameters/tokens. The administrator would need to store this information in the system, so that while accessing that content type, these get passed along for successful access.

Essentially, the administrator would enter this information through Curriculum application. He will also have the ability to test/preview the setup before enabling a provider. Once, the provider is enabled, the Resource Resolver service would then read these settings back and launch the content.

The below diagram shows the workflow:



Learning Tools Interoperability (LTI) service

Introduction

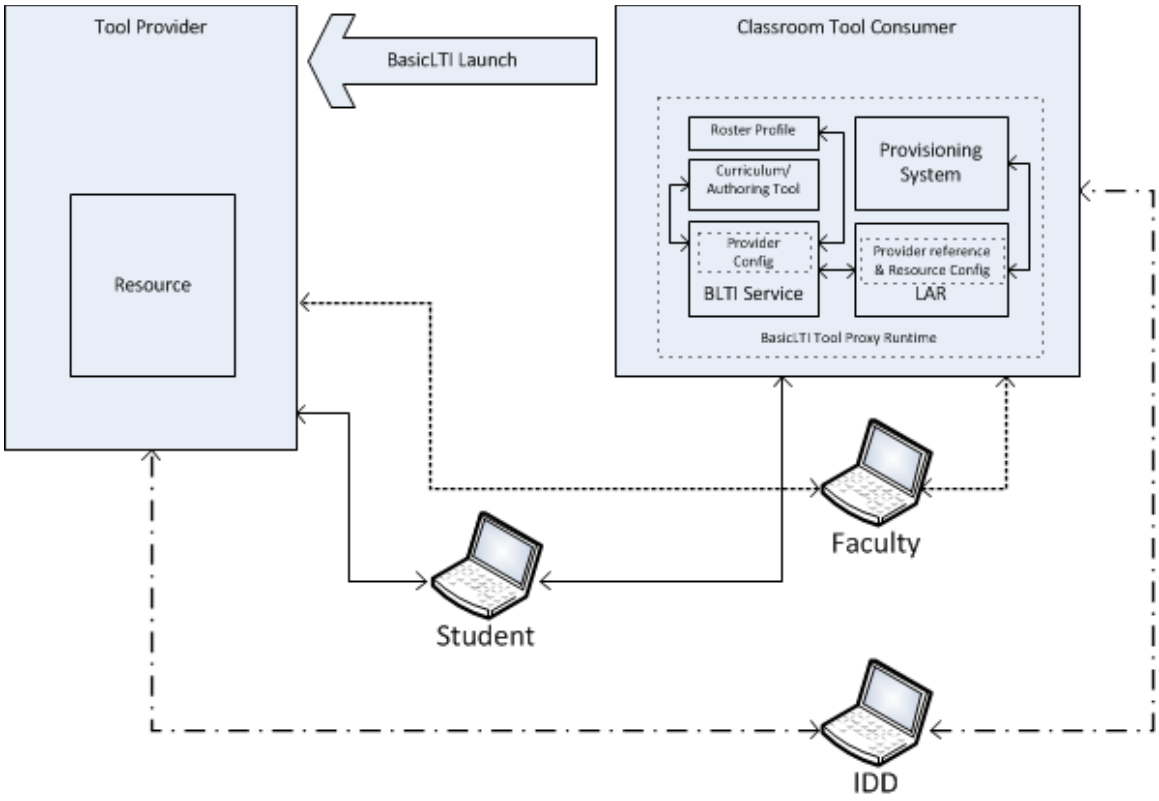
The LTI service is responsible for building the launch form data and send it over to the tool provider.

What does it do?

The LTI service is needed for launching the LTI activities so that the provider gets all the parameters required for launching the content. This is the gateway to connect learning platform to the tool providers.

Architecture

- ✓ Curriculum/Admin tool creates LTI provider configurations for the tool using the content provider configurations.
- ✓ Instructional designer uses Curriculum/Admin application to create an activity and adding provider path along with other activity details.
- ✓ User launches the LTI content from Classroom application.
- ✓ LTI service calls Learning Activity Repository service to get activity that contains details.
- ✓ LTI service creates launch form using provider information, append the information and return the LTI form to Classroom application.
- ✓ Classroom application auto submits the basicLTILaunchFrom and returns LTI tool provider.
- ✓ Basic LTI tool provider negotiates with the Tool Provider and serves the resource.



Format service/Dataviewer

Introduction

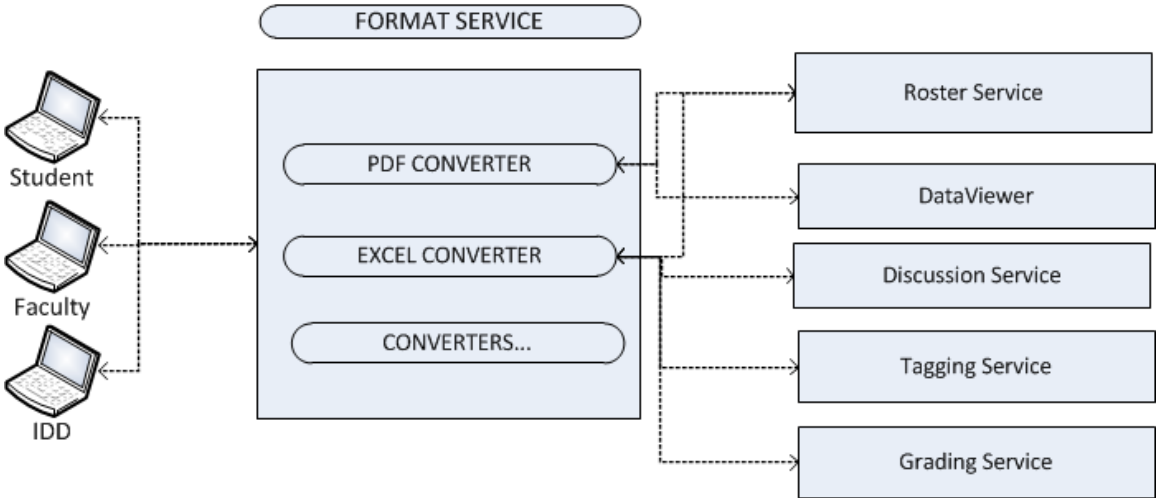
The format service and dataviewer application provide a way for the user to consume data from the learning platform services, in the format that they wish. E.g. PDF, Microsoft spreadsheet etc.

What does it do?

We need the ability for the user to consume the syllabus in printed version in the form of PDF. We also need the ability for the admins or instructors to download the grading, attendance etc reports in spreadsheet format.

Architecture

The service is generic enough to plug-in different formats required to be generated and configured different end points where the data is available. The service takes the parameters as input and the template that has to be used for generating the format. The below diagram shows the overview:



Assessment service

Introduction

Assessment service provides the ability to configure different assessments, quizzes in the learning platform applications. This system allows different questions to be authored and managed centrally and allows consumption of the same through different consumers.

What does it do?

Assessment service provides different capabilities for different users, instructional designers, instructors, students etc.

Assessments Content Management Service

- ✓ This service is used to manage the quiz content in Cognero. It allows users to perform same operations that they would do using Congero UI with APIs.
- ✓ IDDs can perform the following via APIs.
- ✓ Create folders, products, tests and users.
- ✓ Assign the users to products
- ✓ Import tests into Cognero provided the test content is in Cognero XML format.
- ✓ Export tests from Cognero. This gives out the test in Congero XML format.
- ✓ Additionally add individual questions to a test and delete individual questions from a test.
- ✓ Delete tests

Assessments Management Service

The APIs in the service cater to faculty and admin. The operations pertaining to assessment (or quiz assignment) can be managed using this service.

Following services are provided

- ✓ IDD & Faculty can do provisioning of the quizzes.
- ✓ IDD can delete an assignment.
- ✓ IDD & Faculty can change the assignment options
- ✓ Faculty can view quiz results of whole class or for an individual student.
- ✓ Faculty can reset a quiz for a student if there is need be
- ✓ Faculty can get the test or assessment.
- ✓ Faculty can do many more operations.

Assessment Taking Service

- ✓ It provides quiz-taking functionality to students.
- ✓ Students can:
  - Take the quiz via preStart, start, getQuestion, submitAnswer & finish APIs.
  - Check their answers if the assessments allows it.
  - View results of taking the quiz.

Assessments Cron job

- ✓ The assessments cron job auto-submits the quizzes that were not submitted by students if they're timed.
- ✓ It runs every 60 minutes to see if there are any quizzes that were started by students but are not submitted even after the time limit of the quiz is expired. The cron job uses session id of the quiz and uses it to auto-submit the quiz. If the quiz does not have time limits then it does not auto-submit the quiz.

Assessments Analytics & Reporting Service

Analytics involves analyzing the user interaction with learning platform and coming up with usage patterns, then the goal is to present the user with remediation. Currently assessments analytics is limited to presenting the quiz results history to faculty.

Adaptive Learning Engine or Assessments can come up with different usage in the future. The assessments analytics service has 2 components in it:

- ✓ The map-reduce job that periodically pulls data from Hadoop to HBase.
- ✓ The reporting jobs that query HBase and give back report of quiz taking history for a student.

Recommendations Service

Recommendation is a generic service that is used to recommend something to the student. It contains plugins for each use case. The plugin knows what to recommend. Recommendation service will manage or maintain the recommendations in data store.

Assessments service makes use of Recommendation service for following use cases

- ✓ Van Gogh Plugin for Assessments

Van Gogh is self-check for students on Math knowledge. Student will complete the Assessment and based on the results, the assessments service or plugin will

recommend lessons to learn in the Carnegie Learning system (subsidiary of Apollo Group, Inc). Presence of unanswered questions or incorrectly answered questions in each section or unit of the test will result in a lesson being recommended for the unit. Recommendation service will let the student to update the recommendation if he/she has taken the recommendation.

- ✓ A/B quizzes plugin

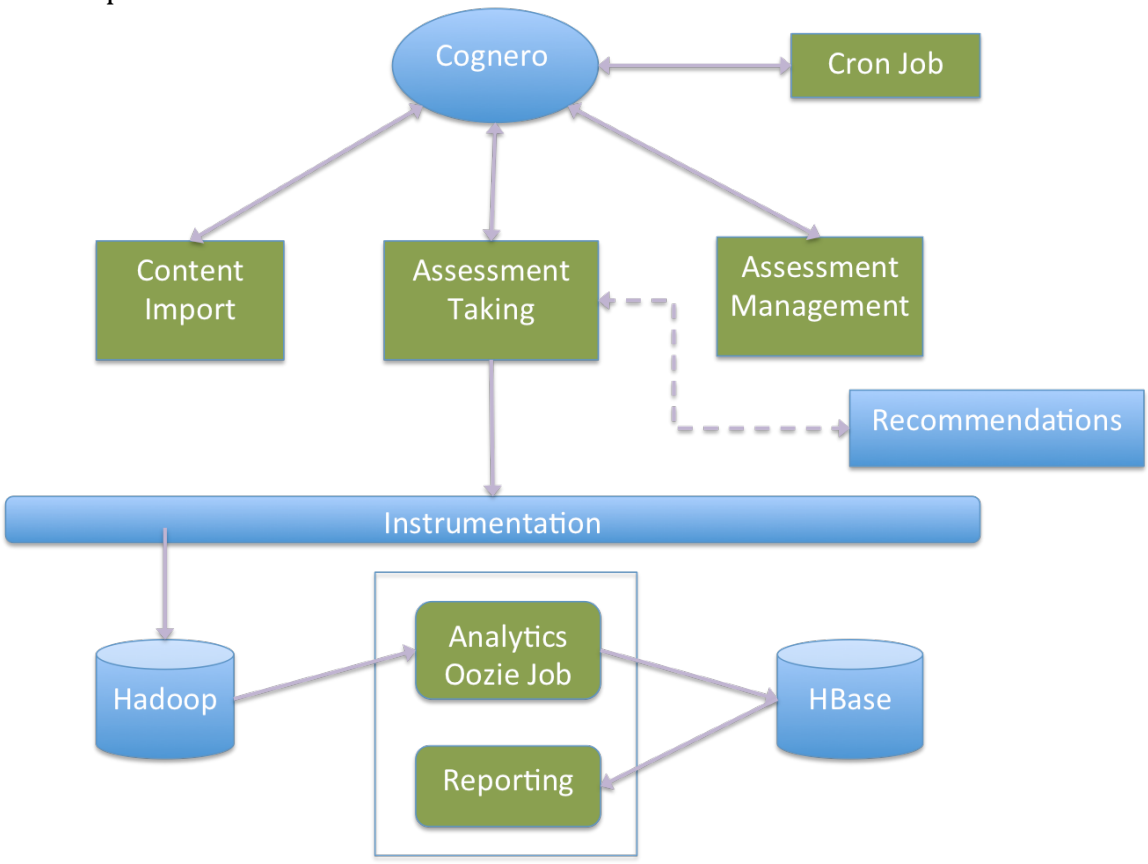
A/B quizzes provide the ability to run variations of a Knowledge Check (KC aka TFA) quiz at any given time. One of these variations will be constant while the other variation will change and overlap. We want the ability to have different classes try different Knowledge Check quizzes.

The assessment service depends on:

- ✓ Roster profile.
- ✓ Authentication.
- ✓ Authorization.

Architecture

Although the assessment service is built on top of Cognero (external vendor), it keeps clear separation between the vendor implementation to the platform consumption.



Assignment submission service

Introduction

Assignment Submission is an integrated and collaborative process in the new classroom that provides an easy means for students to submit an assignment for grading. It provides the student with an easy to use and reliable means of submitting assignments and provides support for a variety of document and media formats.

What does it do?

Assignment submission allows the following features:

- ✓ Individual and team assignment submissions.
- ✓ Personal feedback.
- ✓ Due date alerts (common for all assessment types).
- ✓ Virus detection and removal.
- ✓ Automatic spelling and grammar check.
- ✓ Plagiarism detection.
- ✓ Allows size limitation enforcement.
- ✓ Instructions can be provided to any of the assessment or activity using the Syllabus model and can be applied to assignment upload as well.

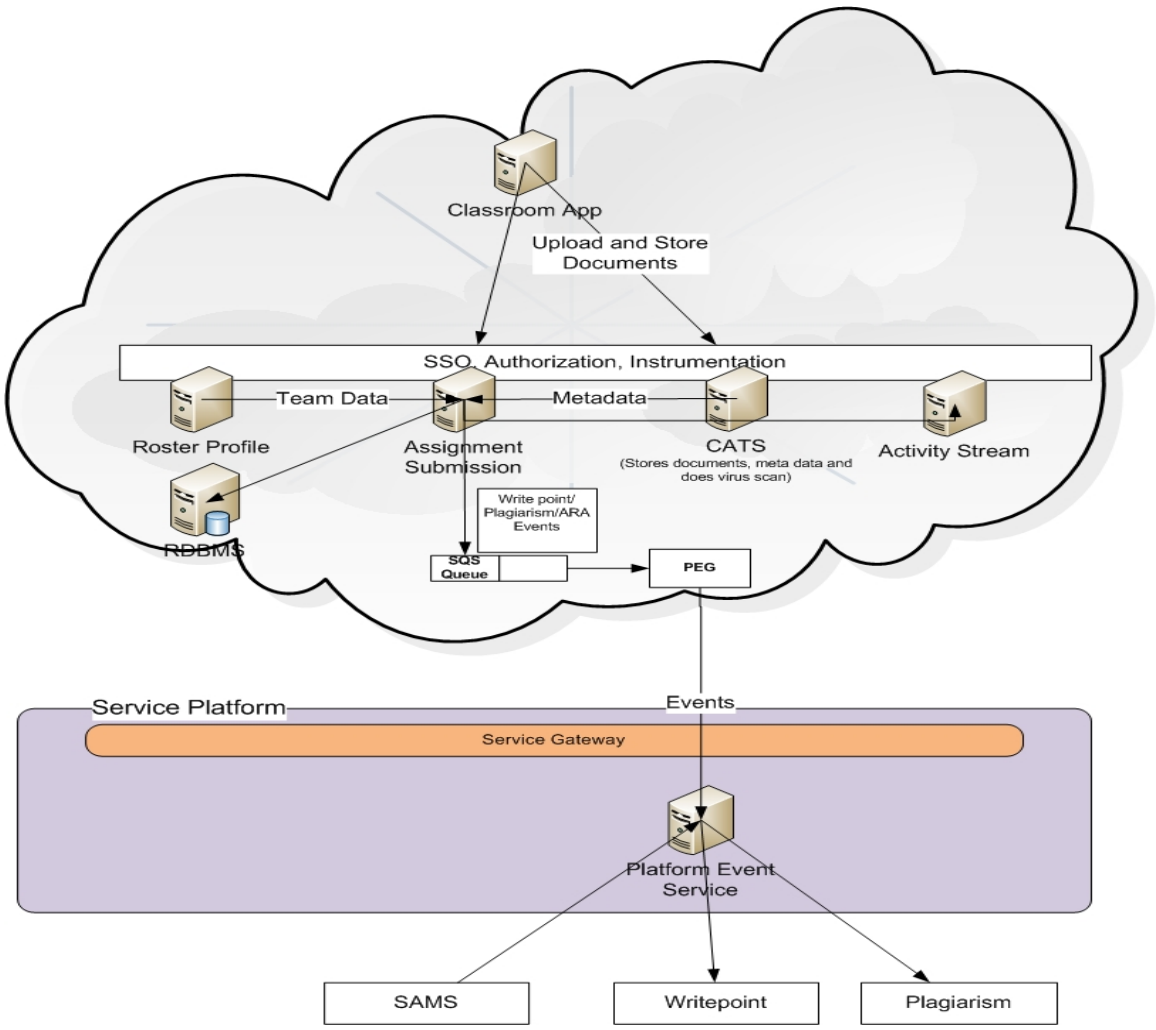
The following user specific use-cases are supported:

- ✓ Students upload documents for Individual/Team assignment.
- ✓ Students make submission.
- ✓ Views submitted documents and submissions.
- ✓ Removes uploaded documents and submissions.
- ✓ Views summary of submissions and faculty feedback in a class/learning unit.
- ✓ Faculty views summary of submissions.
- ✓ Faculty views submissions by assignment for all students/teams in a class.
- ✓ Faculty can download all the assignment for review.
- ✓ Admin can remove a document or submission.

Assignment submission depends on the following components:

- ✓ Integration with CATS (attachment service for document /meta data store and virus scan).
- ✓ Integration with ARA events.
- ✓ Integration with Roster Profile.
- ✓ Integration with Activity Stream.

Architecture



Grade book service

Introduction

Grading is an important component of the learning platform that allows the student submitted assignments to be evaluated by the instructor using the Rubrics system as grading guide, to make sure that the learner meets the outcomes that were set by the instructional designer for the course.

What does it do?

Grading supports the following use-cases:

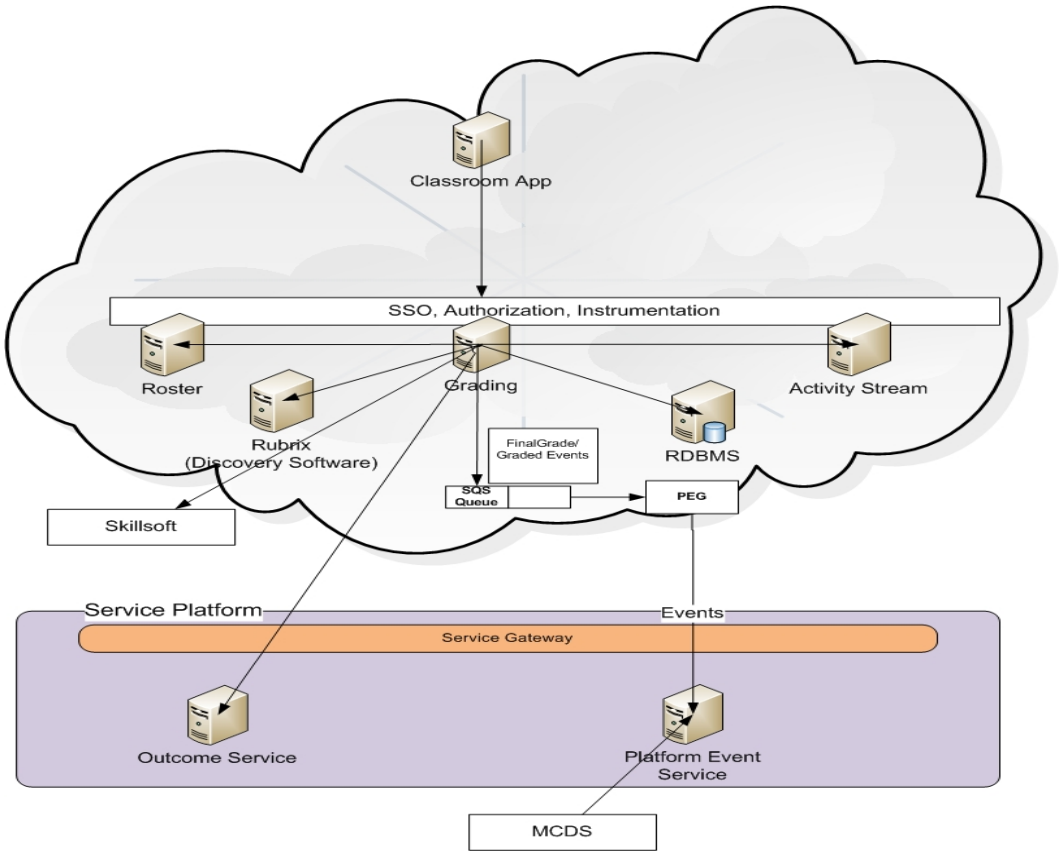
- ✓ Faculty enters score and feedback for a assignment /all assignment in a learning unit.
- ✓ Faculty enters score and feedback for all students for a assignment in a learning unit.

- ✓ Faculty enters score and feedback for all members in a team for a team assignment.
- ✓ Faculty publishes the score and feedback.
- ✓ Faculty/Student/Admin views the published scores and feedback.
- ✓ Faculty imports grade scale from the previously taught course.
- ✓ Faculty edits grade scale for a course
- ✓ Student/Faculty/Admin can view published grade scale.

Grading depends on the following services:

- ✓ Integration with Rubrix.
- ✓ Integration with platform event service.
- ✓ Integration with outcome service.
- ✓ Integration with Skillsoft.
- ✓ Integration with Activity Stream.

Architecture



Rubrics service

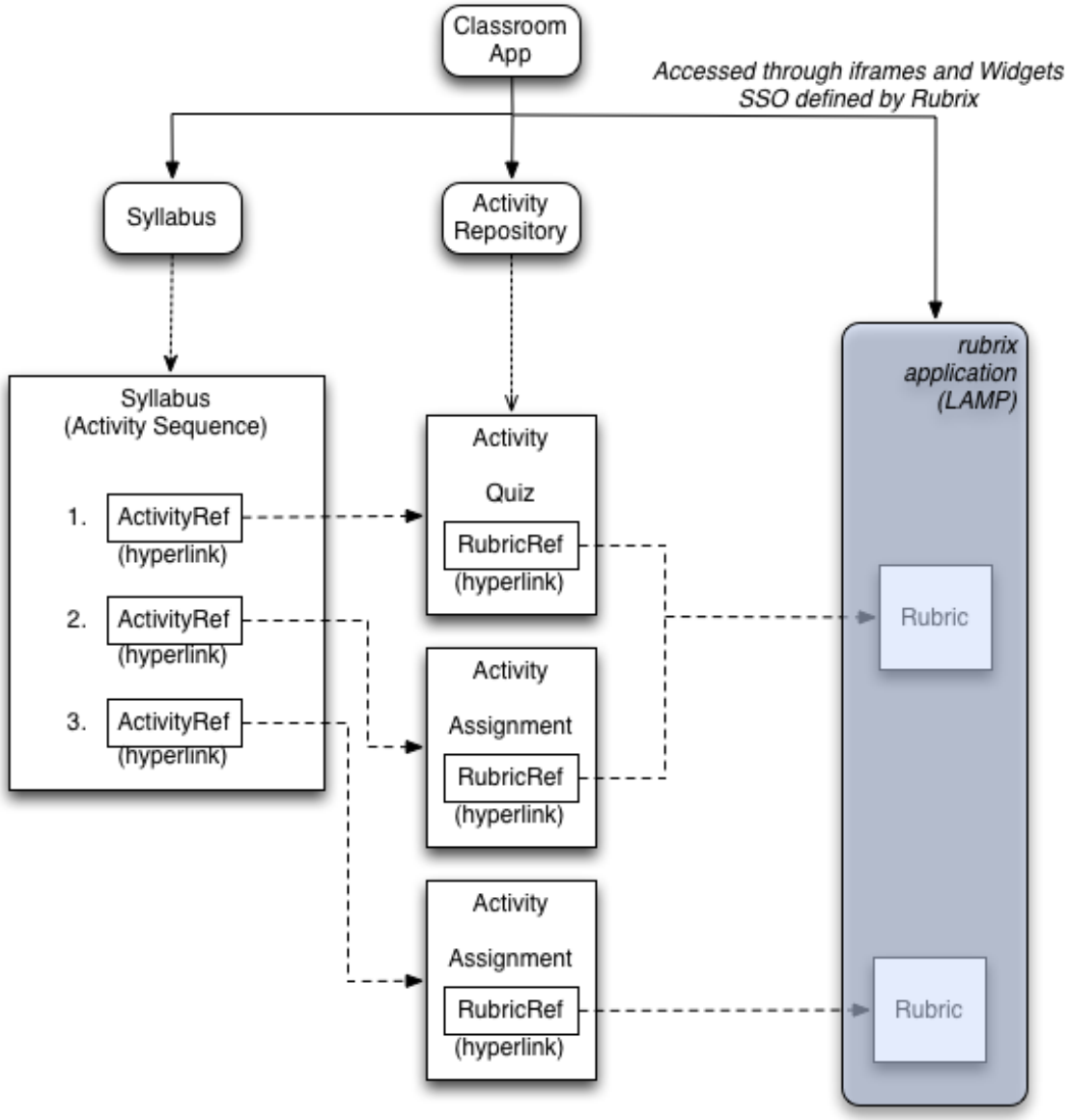
Introduction

Rubric service provides the capability to the instructor a means of communicating expectations for an assignment, providing focused feedback on works in progress, and grading final products.

What does it do?

- The instructional designer would be able to pick the rubrics from the listing and assign it against an assignment.
- The instructor will be able to look at the Rubrics, score the Rubric.
-

Architecture



Meta data service

Introduction

Meta data service is based on the Resource Description Framework (RDF), which is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources. RDF

metadata can be used in a variety of application areas; for example: in resource discovery to provide better search engine capabilities; in cataloging for describing the content and content relationships available at a particular Web site, page, or digital library; by intelligent software agents to facilitate knowledge sharing and exchange; in content rating; in describing collections of pages that represent a single logical "document"; for describing intellectual property rights of Web pages, and in many others. RDF with digital signatures will be key to building the "Web of Trust" for electronic commerce, collaboration, and other applications.

An extensible metadata service should be able to manage metadata for any resource and should be able to support a wide variety of metadata models. Any metadata service needs to be extensible for it to be useful. Metadata standards such as IEEE LOM (Learning Object Metadata) and Dublin Core (DC) have been prescriptive in their approach. They have sought to adopt a more-or-less standardized set of elements with prescriptive semantics for those elements. While such standards have been adapted to a reasonable extent, interoperability between implementations of this standard is not easy (since semantics are not precise). IEEE LOM is specific to learning and has limited applicability beyond the domain of learning. DC is more general purpose but supports a limited set of attributes and cannot model domain specific metadata.

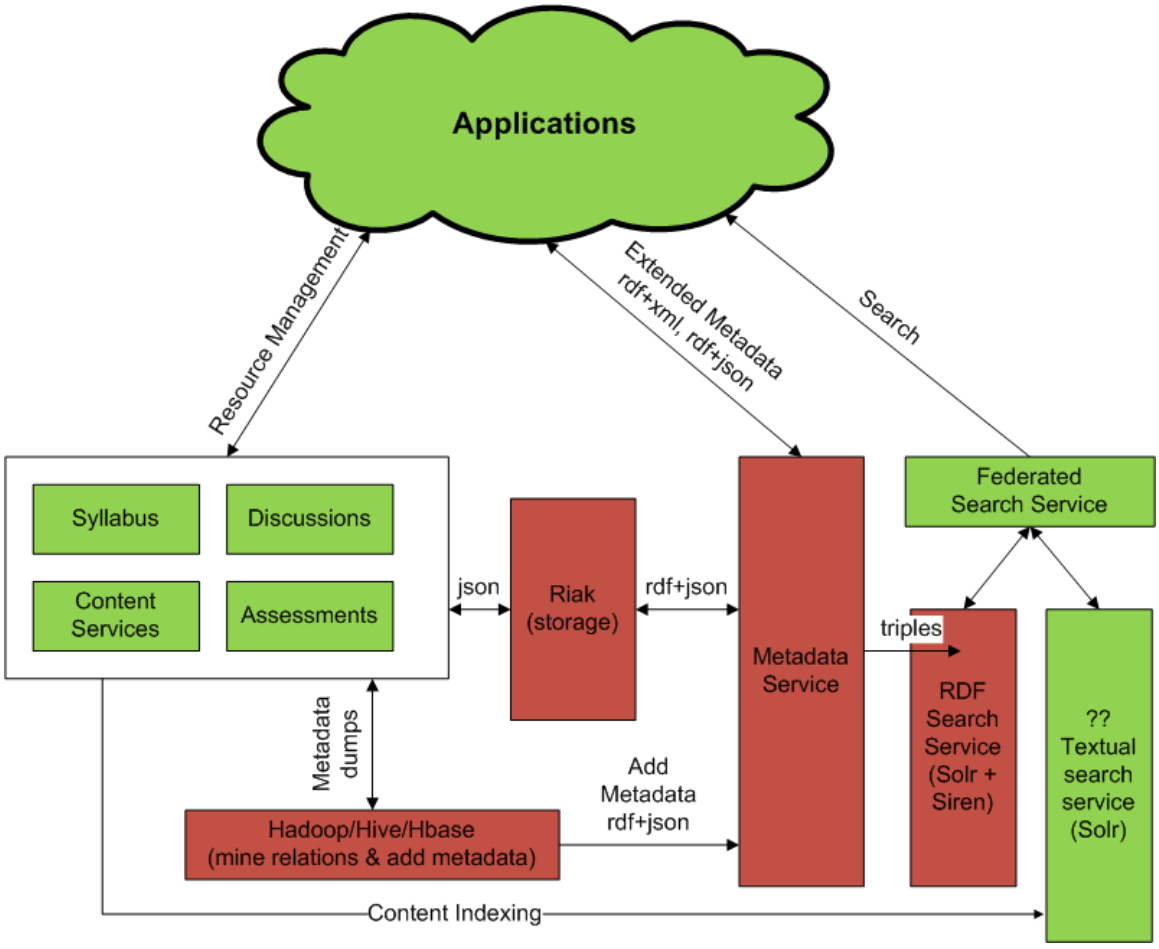
Other metadata standards such as RDF have been designed around the ability to model any metadata for any resource while maintaining strict semantics. At its core, RDF uses a very simple { Subject, Predicate, Object } model to represent metadata. Thus an RDF for a resource is a directed graph such that the target (i.e. Object) of any link as well as the link (i.e. Predicate) can be defined with strict semantics. RDF leverages existing standards such as Dublin Core (DC) and Friend-Of-A-Friend (FOAF) to identify predicates thus providing the ability to interoperate with these metadata standards. RDF also provides query languages to query resources based on their metadata. RDF is designed to support a distributed resource model where each resource manages its own metadata while still linking with other resources. Resources could exist anywhere in the World-Wide-Web.

What does it do?

- ✓ An "author" wants to categorize a media object so Classroom can surface the media object in context based on
  - The school the student belongs too
  - The student's interests
  - The topic that is being "promoted" within Classroom, etc
- ✓ A faculty wants to select an activity from previous syllabi that enables a student to learn a given objective in a given course. The learning activity could be a discussion, quiz or a learning object

- ✓ A student has answered a question incorrectly and ALE needs to understand the concepts being tested by the question so it can determine if and how to remediate.
- ✓ A faculty/ALE needs to find remedial content to help a student understand a concept. The content needs to match the concept and must be at the right level of instruction.
- ✓ IDD is designing a new course and needs to find quizzes and content that are relevant to a given topic, Bloom's level, etc
- ✓ These use-cases surface metadata needs on various content/learning objects. Given that a content object could become a useful learning object and vice-versa, it is best to think of content and learning objects as being similar and manage them uniformly (i.e. in the same set of repositories).

Architecture



Discussion service

Introduction

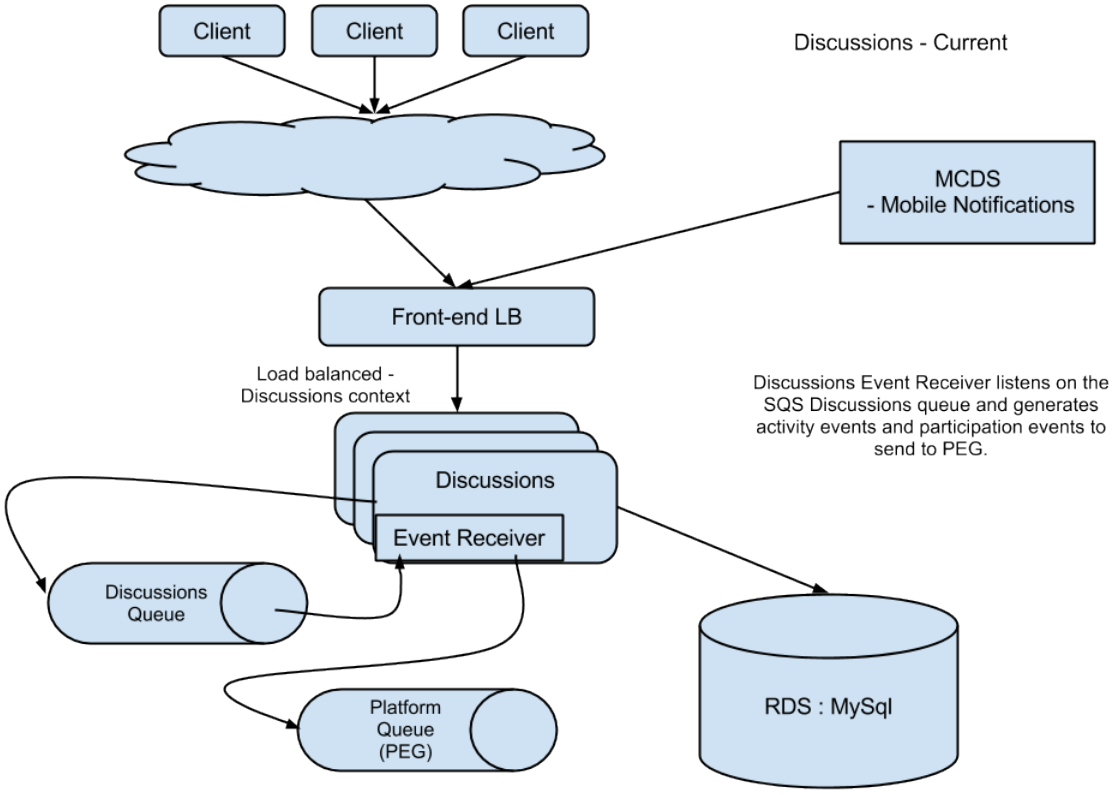
Discussions service provides the capability to post and read messages between authenticated users within a classroom context.

What does it do?

Currently, there are five types of discussions that are supported:

- ✓ AnnouncementDiscussion - Used to allow faculty to communicate announcements to all students enrolled in a class. Only faculty can post announcements.
- ✓ AssignmentDiscussion - Used as assignment discussion questions.
- ✓ CommentDiscussion - Used to allow comments on learning activities.
- ✓ GeneralDiscussion - Used for general class discussions.
- ✓ PrivateDiscussion - Used for private conversations between faculty and student.

Architecture



Attachment service

Introduction

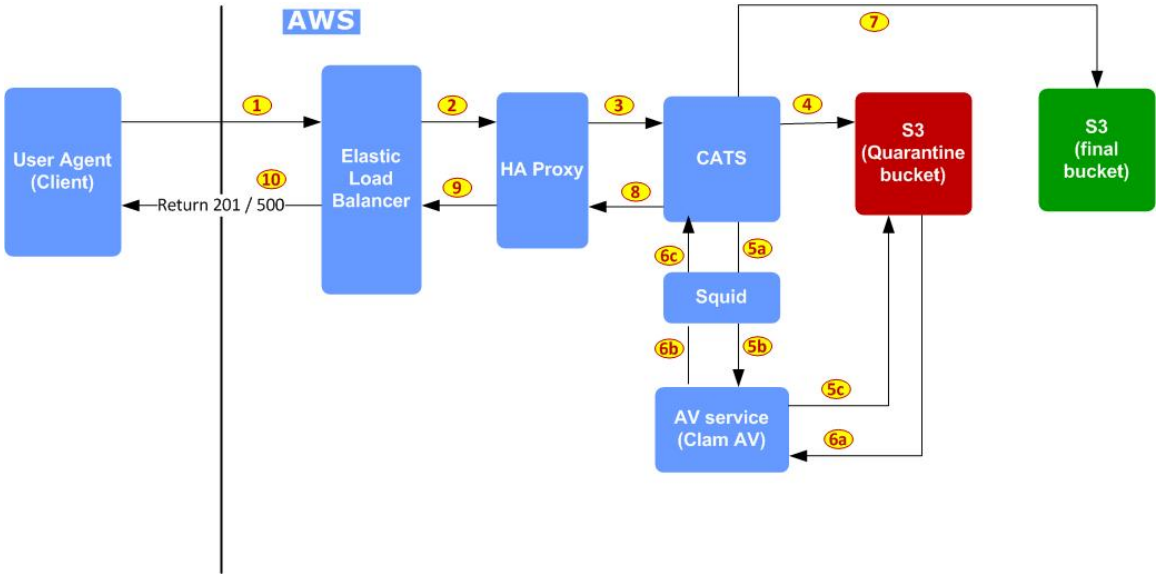
The learning platform applications such as assignment upload and discussion components require service for uploading the attachments, provide metadata around the attachment, sanitize them and consume them back in different use-cases. The attachment service provides a mechanism for meeting these requirements.

What does it do?

Attachment service provides the following use-cases:

- ✓ Uploading of the attachments of different content types both using the HTML5 chunked model and regular form based model.
- ✓ Managing the meta-data for the attachment.
- ✓ Scan for viruses for the files and quarantine them where possible.
- ✓ Provide mechanism to download and view the same.

Architecture



Content pipeline

Introduction

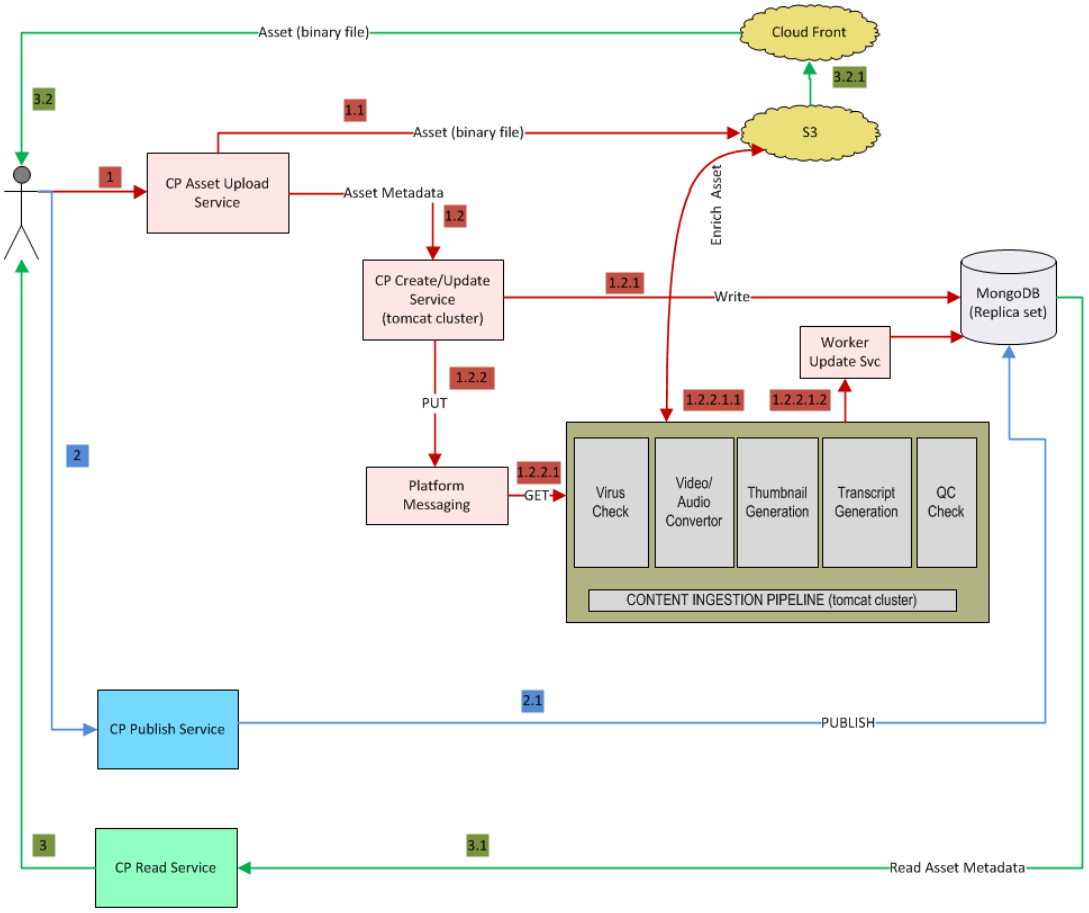
As the name indicates, Content pipeline is a central repository for managing the content and providing a uniform experience across different systems.

What does it do?

Content pipeline supports the following use-cases at a high level:

- ✓ Content ingestion.
- ✓ Content metadata storage.
- ✓ Content metadata management.
- ✓ Version control.
- ✓ Application and API level delivery of content services.

Architecture



Applications overview

Classroom

Classroom application is a rich, multi-tenant, polymorphic, Internet application that is the home for the students, instructor and admin users and is built using the client development framework. It is also a Launchpad for several tools and contents that serve the needs of the learner.

Classroom application is the component that is the entry point for all the learner interaction. This is highly customizable, configurable and polymorphic application. The application allows very granular level of control for turning on/off a particular module or a feature as a whole, using the Access Manager component.

In a way the application can be considered as a reference implementation for showcasing all the capabilities that the learning platform provides. Although, the learning platform applications are built using the client development framework, any new modules that are built either with or without using the client development framework can easily be integrated.

Classroom application is the home for the “Magic Box” that provides a uniform experience for the users to consume any content.

Classroom application is polymorphic, in that it provides different experience for the students, instructor, instructional designers and admin users based on the applicable context.

Classroom is built as an aggregation framework using the following modular components that could also be consumed independently:

- Activity stream: The activity stream (patented) provides a consolidated view of all the activities that is going on in the Classroom. It has the ability to automatically update the notification when a new activity occurs in the class. It also provides the ability to filter in different ways.
- Discussion: The discussion system provides the discussion capability through both as a full-blown learning activity and as an integrated commenting section against each learning activity. The discussion project also provides the capability for the instructor to post the news module. It also provides tagging, replying, editing capabilities for the comments or discussions. The discussions could also be controlled by depth of replies.
- Quiz: The quiz project is the Front-End application for the assessment system that provides the ability to take the assessment (both quiz and final exam types) types of learning activity.
- Data viewer: The data viewer project is the Front-End for the format service. This project provides the view or the display component including the display logic for the PDF version of the syllabus and similar components.
- Grading: The grading project provides the ability for the instructor to assign grading for the assignment activities. It also provides the grade scale and Rubrics that complement the instructor’s input.
- Assignment upload: The assignment upload project provides the ability for the student to submit the file upload type of assignment.
- Admin capability: The admin dashboard project provides the ability for the administrator to view and verify the activities in the class. The example capabilities include the resetting an exam, viewing the attendance events etc.

The following are few of the capabilities of the Classroom application:

- ✓ Showing a home page that contains a consolidated view of important things in the class for the user. This includes:
  - ToDo carousel that shows the context-sensitive, consolidated and personalized experience to the user.
  - Roster view that shows the ability for the user to connect with his team and other learners in the class.

    - News module that shows important announcements about the class for the learner.
    - Activity stream that shows the consolidated view of all the relevant and personalized activities in the class.
    - Class switching option that allows the user to quickly toggle between different classes that he is enrolled/assigned.
- ✓ Syllabus view that shows the entire syllabus with the linked objectives, grouped by required, recommended activities and assignments. The same view can be filtered either by week or by assignments alone too. The individual learning unit could also be collapsed or left expanded and the state resumed between sessions.
- ✓ Activity detail view (aka Magic Box) that allows consuming the content in a uniform way without having to install additional plugins. It also allows the ability for the size of the display to be adjusted, the whole window to be expanded, popped out into its own instance.
- ✓ The application is resilient to individual service failures. Unless one of the bootstrapper (core dependency) services fails, the application handles the failures fairly contained, allowing the user to interact with the other parts of the application successfully.
- ✓ The application takes polymorphic approach in that it supports different users:
    - For the Instructional designers (IDD) the Classroom application shows the preview using the same set of tools that would be used to render for the student.
    - For the instructors:
        - The application shows a sandbox mode, where the instructor could create his version of the course template and preview the same.
        - The application shows a syllabus-editing mode, where the instructor could create his version of the syllabus from the CDG template or his versions of the templates and publish them to make it available to the students.
        - The application takes into course offering mode, where the instructor can interact with the students, grade the assignments, moderate a discussion thread, tag a discussion post etc.
    - For the students:
        - Before the class begins, the application shows a CDG view of the course, where the student can see the static content part configured in the syllabus and get familiarized with it.
        - Once the class begins, the applications shows a course offering mode, where the student takes the class. He will be able to consume all the content that are available in the course, complete the assignment etc.
- ✓ The application is multi-tenant capable. The same instance deployed is capable of delivering:

- Different types of activities/tools based on tenant.
- Different messages based on tenant.
- Different theming (including CSS, images etc).
- Different learning modes.

✓ Failure tolerance or service resiliency.
✓ Micro level interaction details captured (deeply instrumented).

Curriculum:

The Curriculum application allows the instructional designers and course publishers to author the course, create the content and maintain the versions. Curriculum application is the flip side of the coin for the Classroom application i.e. it allows authoring or producing the course and content that the Classroom would consume. Both these applications make use of majority of same set of shared services.

It supports the following use-cases:

- ✓ This is the step where the instructional designer would decide the general objective for a course, details for the course and creates the activities and assignments that would help the student learner to meet the objectives.
- ✓ The instructional designer could reference the contents that are available in external URL, internal content pipeline, upload contents; create upload assignments, quizzes, discussions etc.
- ✓ This is also the step where the tenant admin will incorporate external vendor integration such as Learning Tools Interoperability (LTI) tools.
- ✓ Once after the vendor integration is added, the instructional designer can then create a content mapping for the vendors.
- ✓ The instructional designer could create pre-start learning unit, where he can indicate what should be done before the class starts.
- ✓ The instructional designer could version the course that is being taught.

Knowledge Check:

We are developing a world-class online classroom environment, enabling students, faculty and the institution to connect, collaborate and learn in an always on, robust and performant environment. Many individual components make up this complex workspace, and to ensure our students receive a truly individualized learning experience, we must deliver on a companion platform that not only collects the data and signals generated from the online activity, but also analyzes it for patterns and elicits recommendations for individuals that can be served in a variety of ways and at different intervals. Essentially, ALE (Adaptive Learning Engine) must deliver on the idea of serving the *right student with the right data at the right time*.

ALE Knowledge Check (aka KC aka TFA) application was built as one component of the ALE ecosystem to deliver a tutored experience to the students while assessing the depth of their knowledge on different concepts taught in their course. The goal is to help students learn the "why" when they struggle with an exercise and provide them support to correct their misconceptions in a timely manner.

The Adaptive Learning Engine is a platform developed to enhance the teaching & learning environment we're developing. It is an intelligent learning system that makes personalized recommendations by analyzing a student's performance. It continually assesses a learner's style, struggles and strengths and keeps the student on track by guiding them in the right direction at the right time.

The Knowledge Check application presents relevant eBook/external content to students when they have misconceptions about a particular question and then allows them to reattempt the question in order to provide the student a way to ascertain if the content has helped them clear their misconception. After getting the question correct, a description of why the chosen answer was correct is given so that the student can validate their reasoning while answering a question. After submitting the quiz, a personalized study guide is presented to the student, which breaks down their assessment performance into concepts and gives them relevant content to further their knowledge. The personalized study guide is available to the students throughout the duration of their course.

The Knowledge Check faculty dashboard provides the faculty a few different views into the performance of their class or individual students. It also allows the faculty to send additional content and/or personalized messages to students relating to their Knowledge Check.

The Data Science also analyzes the data generated from the Knowledge Check applications and IDD (Instruction Design and Development) teams to determine the efficacy of quiz question, hints, reading materials etc. Based on this analysis, the IDD organization works to improve the Knowledge Checks in order to provide the best possible learning experience to the students while assisting the faculty members by providing them the most relevant data about their students' performance.

The application supports following use-cases:

Student Facing Application

- ✓ Allow students to take weekly quizzes
- ✓ Provide students hints before attempting to answer a question
- ✓ Give the user feedback on their selected answer

- ✓ If the student answer was correct, display explain as to why the answer choice was correct.
- ✓ If the student answer was incorrect, display reading/audio-visual content.
- ✓ Allow student to reattempt the question (maximum of 2 attempts per question) if the first attempt was incorrect.
- ✓ Give the user feedback after the second attempt and regardless of the correctness of the answer choice, explain what was the correct answer choice and why.
- ✓ Let the user close their session at any time and resume of where they left off later.
- ✓ Display their personalized study guide after submitting their Knowledge Check.
- ✓ Send a message to the student with a link to their personalized study guide, which can be accessed throughout the duration of their course.

Faculty Facing Application:

- ✓ Present student performance data to the faculty for their class broken down by concepts.
- ✓ Allow the faculty to view detailed report about a particular student's performance.
- ✓ Allow the faculty to send additional content/comments to one of more students.
- ✓ Allow the faculty to reset a student's Knowledge Check in order to let them take it again after submitting it once.

Data Collection:

- ✓ Instrument actions performed by the students and faculty so that the data science team can analyze the efficacy of different sets of data presented to the users.

Admin console:

The admin console is an application that unifies several of the tenant administrator tasks. This is the place where the administrator starts setting up the users, roles, and courses to be taught, configurations for the applications etc. He will also be performing the operational administrative tasks such as monitoring student/faculty attendance, moderating discussions, resetting exams, handling the access to the class, publishing the syllabus if it is not already published and similar tasks.

Mobile platform overview

The mobile platform is built to support the learning experience on the mobile and tablet devices. The main differentiating factor for the mobile platform from the regular browser based applications is that it incorporates native application

development to smoothen the user experience and make extensive usage of the device capabilities. This means that although the same set of ReST based web services are consumed, the amount of data delivered needs to be tuned to be optimal for the mobile devices and networks. The content delivery mechanism also needs to be adjusted to meet the mobile platform capabilities. This also means that the notification capability present on these devices is made use of to instantly alert the student about an occurrence of the event that is applicable to the user.

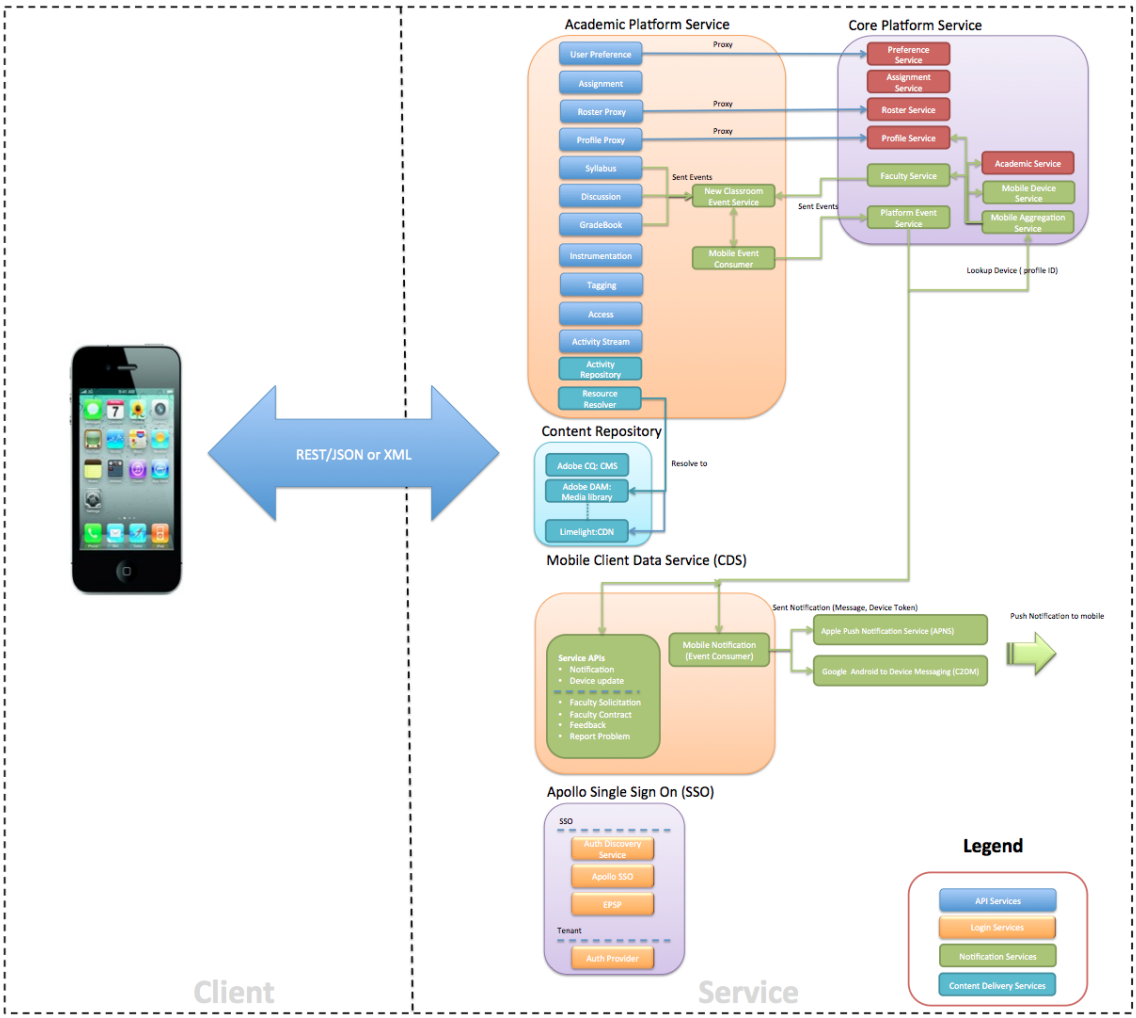
Mobile platform supports the following platforms:

- iOS
- Android

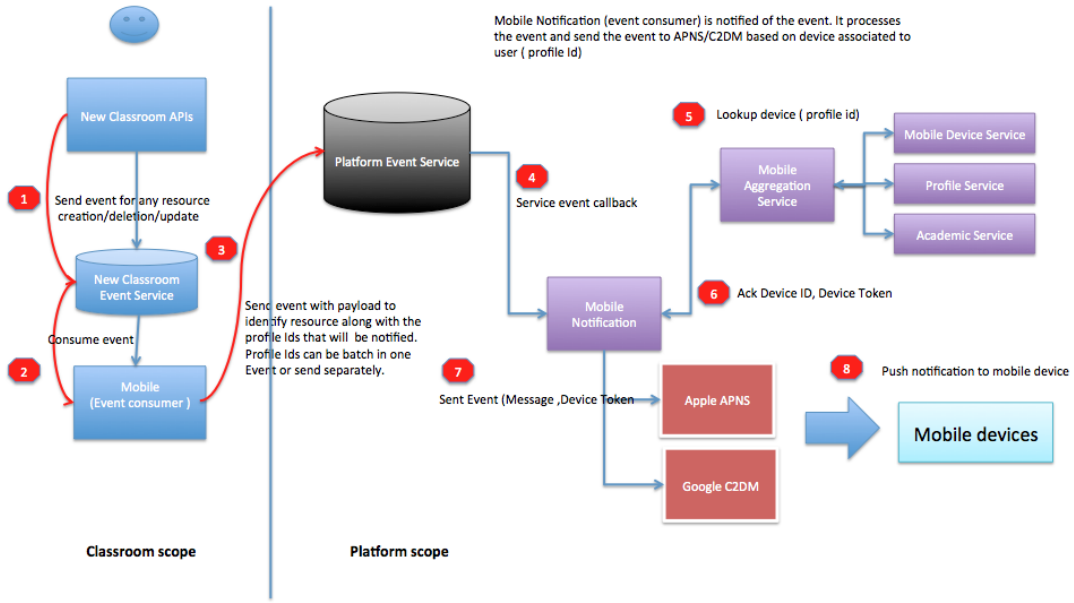
The following use-cases are supported

- ✓ Device access - Available for iOS and Android smartphones.
- ✓ Student and Faculty support.
- ✓ Access to current and previous classes.
- ✓ Class materials and assignments - Students are able to access learning activities and assignments and mark completed activities. Only supported materials will be displayed (ebooks, videos, podcast, pdf, word, excel, power point).
- ✓ Class Discussions - Students and faculty are able to browse and reply to classroom discussions and create new messages.
- ✓ Class Participation - Students are able to track their participation marks. Faculty is able to mark student's posts for participation requirements.
- ✓ Syllabus - User is able to view complete syllabus for the course per learning module.
- ✓ Class Announcements and Grades - Student is able to browse class announcements posted by faculty. Faculty is able to post, edit, and remove announcements.
- ✓ Notifications - Provide real time student and faculty alerts triggered by activity in the classroom (new messages, grades, replies to posts, etc)
- ✓ Offline access - Classroom data is made available to user in offline mode (browsing posts, announcements, syllabus, draft messages and class materials)

Mobile platform high-level overview


Academic Platform Service
Core Platform Service
User Preference
Assignment
Roster Proxy
Profile Proxy
Syllabus
Discussion
GradeBook
Instrumentation
Tagging
Access
Activity Stream
Activity Repository
Resource Resolver
Proxy
Proxy
Proxy
Sent Events
New Classroom Event Service
Mobile Event Consumer
Preference Service
Assignment Service
Roster Service
Profile Service
Faculty Service
Platform Event Service
Academic Service
Mobile Device Service
Mobile Aggregation Service
Sent Events
Lookup Device (profile ID)
REST/JSON or XML
Content Repository
Adobe CQ: CMS
Adobe DAM: Media library
Limelight:CDN
Resolve to
Mobile Client Data Service (CDS)
Service APIs
Notification
Device update
Faculty Solicitation
Faculty Contract
Feedback
Report Problem
Mobile Notification (Event Consumer)
Sent Notification (Message, Device Token)
Apple Push Notification Service (APNS)
Google Android to Device Messaging (C2DM)
Push Notification to mobile
Apollo Single Sign On (SSO)
SSO
Auth Discovery Service
Apollo SSO
EPSP
Tenant
Auth Provider
Legend
API Services
Login Services
Notification Services
Content Delivery Services
Client
Service

Notification flow



Technology stack

- Languages used
- Frameworks used
    - Platform
    - Services
    - Applications
- Persistence layer.

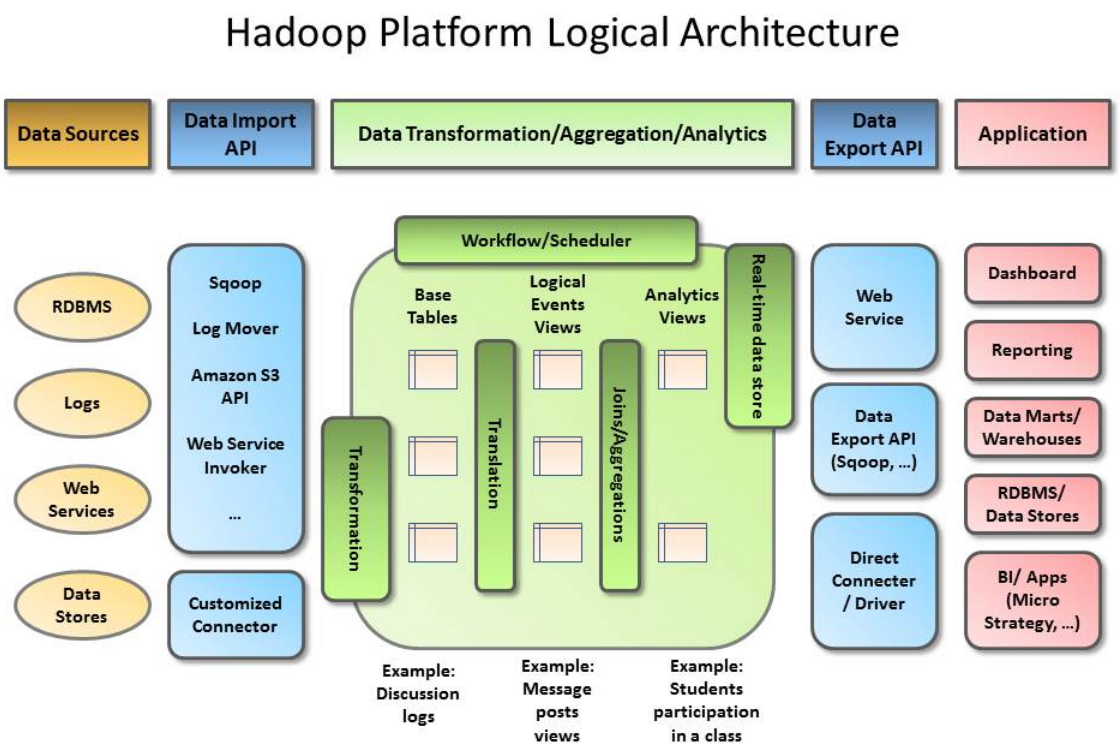
Function	Technology	Comments
REST	HTTP	
Client/FE	HTML, CSS, JavaScript, GWT	
Client/FE, Learning Tools	BLTI	Uses OAuth, extended specific to learning
Services	JDK, J2EE, Jersey, OSGi, Glassfish	All server-side component interactions are enabled through OSGi services in a Glassfish *learning platform container*
Persistent Store	Riak	Schema-free model suitable for rapid feature additions, horizontally scalable
	MySQL (RDS)	Relational model, rapid development

	S3	Web services based file storage
	MSSQL	Storage layer of 3rd party solution (Cognero)
Cache	Riak	Simple key-value, distributed hash table
Events, Asynchronous Messaging	SQS	Simple web services interface, runs on cloud; traditional solutions do not work well on cloud; runs in same environment as other classroom components
Assessments	Cognero	
RDF Triple store	Allegro Graph	
Search	SOLR	
XML Store	MarkLogic	
3rd party integrations	SAML	

Appendix

Additional detailed information for the components.

Appendix-A (Data pipeline overview)



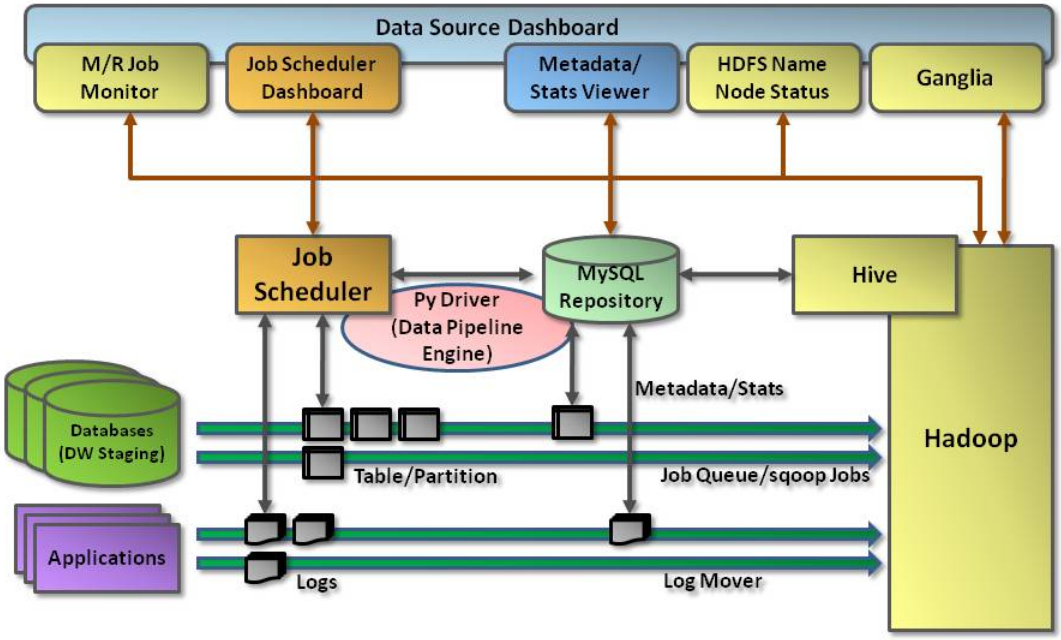
Production Hadoop cluster:

- 30 nodes with 150 TB (70 TB in use as of Aug 2012) total raw storage capacity
- Scaled down Staging cluster (12-node) and QA cluster (8-node)
- Virtual Machines for development environment
- All software stacks are open source (Hadoop, Hive, Hbase, Oozie, Sqoop): the Hadoop "Ecosystem"
- Support contract with Cloudera

Data Pipeline:

- Infrastructure to bring in data from databases, applications, and other sources
- Data import API layer (Sqoop, Log mover, Amazon S3 downloader)
- Workflow/Scheduler (Open Source Job Scheduler, Oozie)
- Batched query-enabled infrastructure (Hive)
- Real-time (sub-second) data store (HBase)
- Monitoring/Alerting/Trending (Nagios, Ganglia)
- Data export API (Sqoop, Web Services) and direct cccess (Tableau connector, MicroStrategy connector)

Data Pipeline Architecture Diagram



The above diagram shows the high level architecture of Data Pipeline. At the end of the release, 2 main pipelines will be built:

Database pipeline - Since all databases that we are interested in will be replicated to Staging, the database pipeline will be focusing on extracting data from Staging Oracle and loading to the Hadoop cluster. The database pipeline will be based on "sqoop" technology (open source from Cloudera).

Application pipeline - "Log mover" will be deployed on Hadoop gateway server(s) to pull data from different application servers.

Main Components:

- Job Scheduler - We will deploy "open source job scheduler" as our job scheduler solution.
- Job Dashboard - We will use the GUI/dashboard from open source job scheduler for this release
- Py Driver - Py Driver is developed by the Data Pipeline engineer team and is used as the execution engine for Data Pipeline

MySQL - MySQL will be used for several purposes:

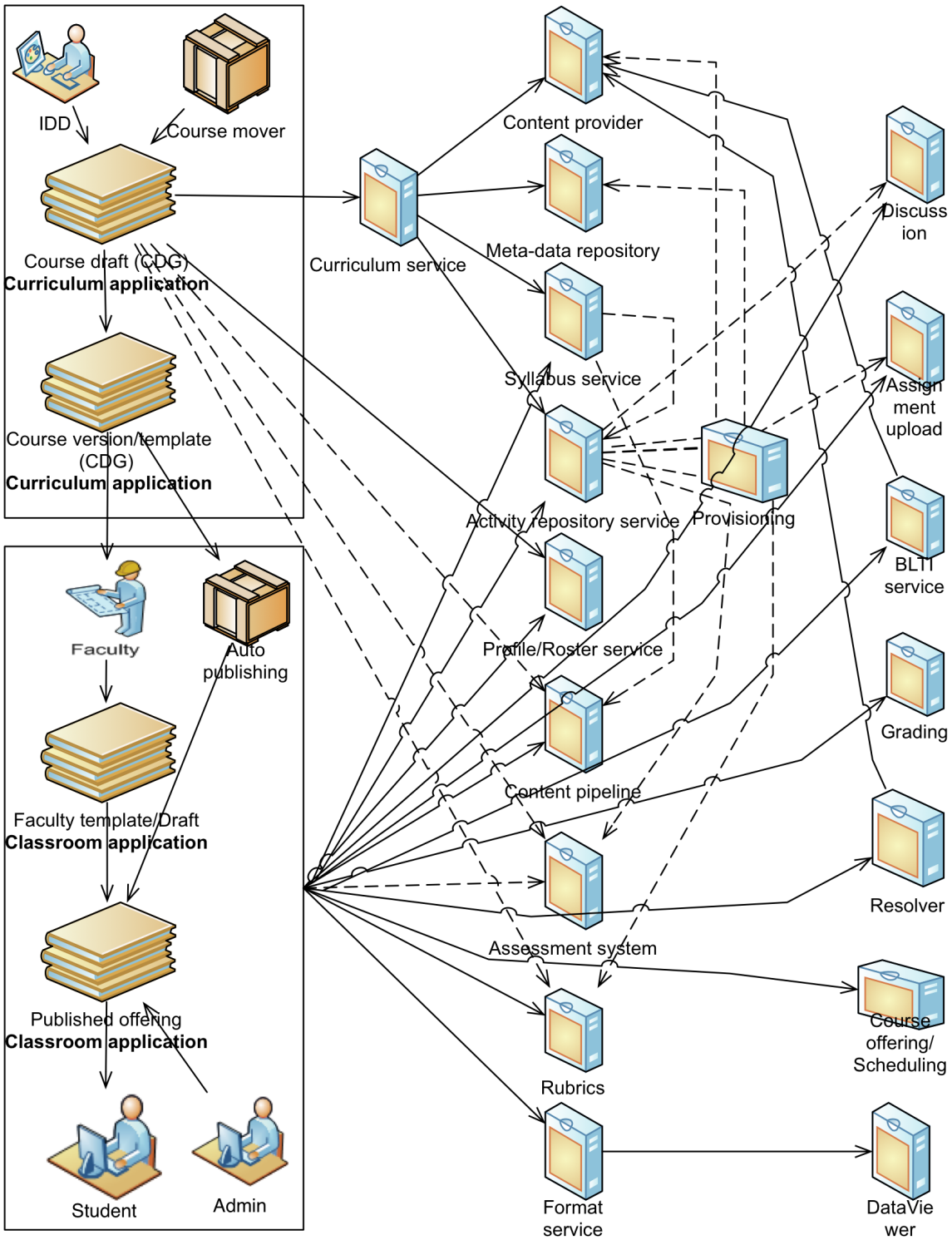
- Storing metadata for Oracle Staging tables
- Storing metadata for Hive tables
- Storing stats for sqoop execution
- Storing stats for log mover operations

Hive - Open source Data Warehouse infrastructure on Hadoop.

Data Source Dashboard - Data Source Dashboard has been developed by the Data Pipeline team to achieve the following requirement:

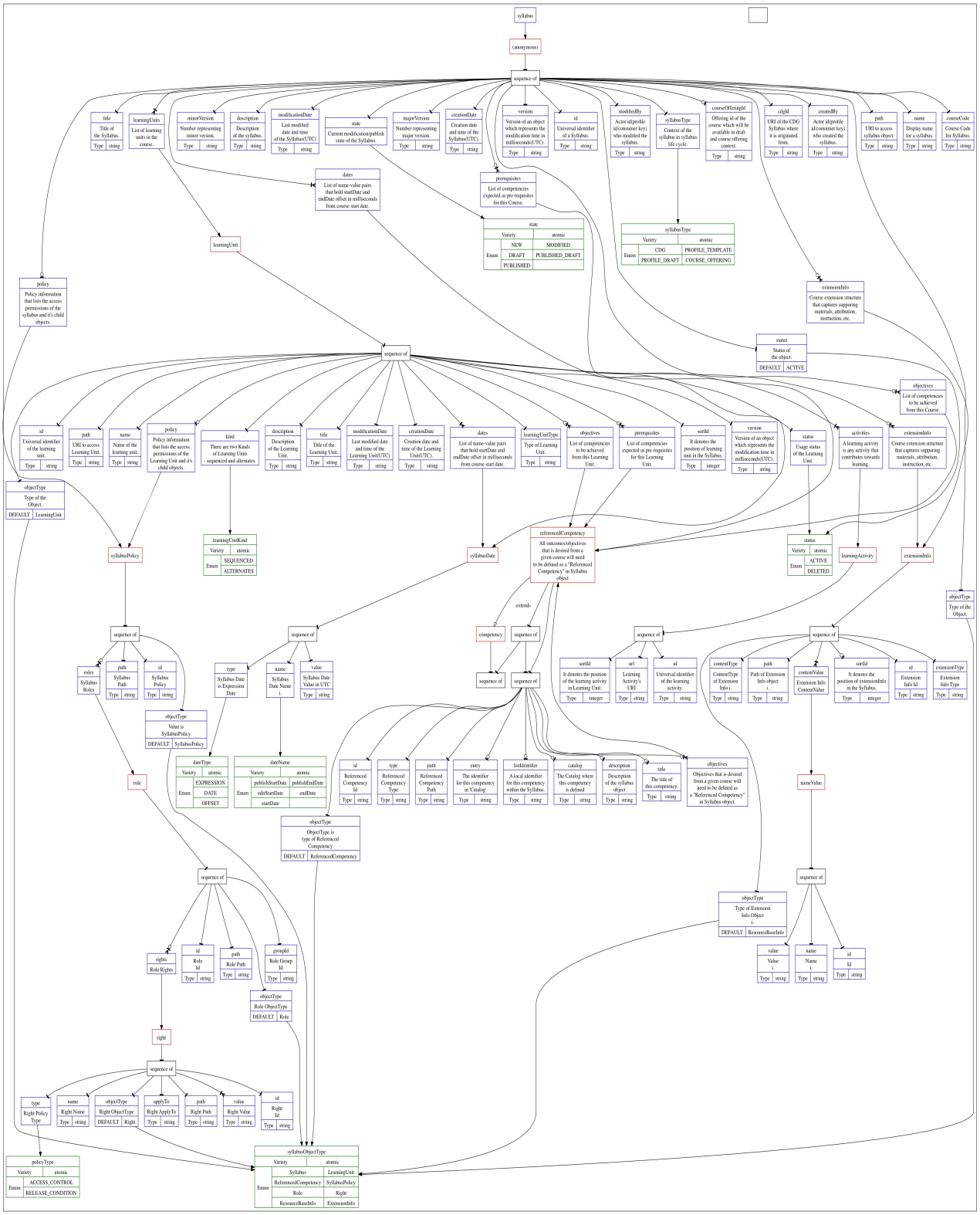
- Current data source in the pipeline can be viewed.
- Available data source (from Staging or from log movers) can be populated on the dashboard.
- A data source can be included in the data pipeline from the dashboard if not yet included (via Shopping cart or other similar feature).

Appendix-B (Overall workflow and system interconnection)

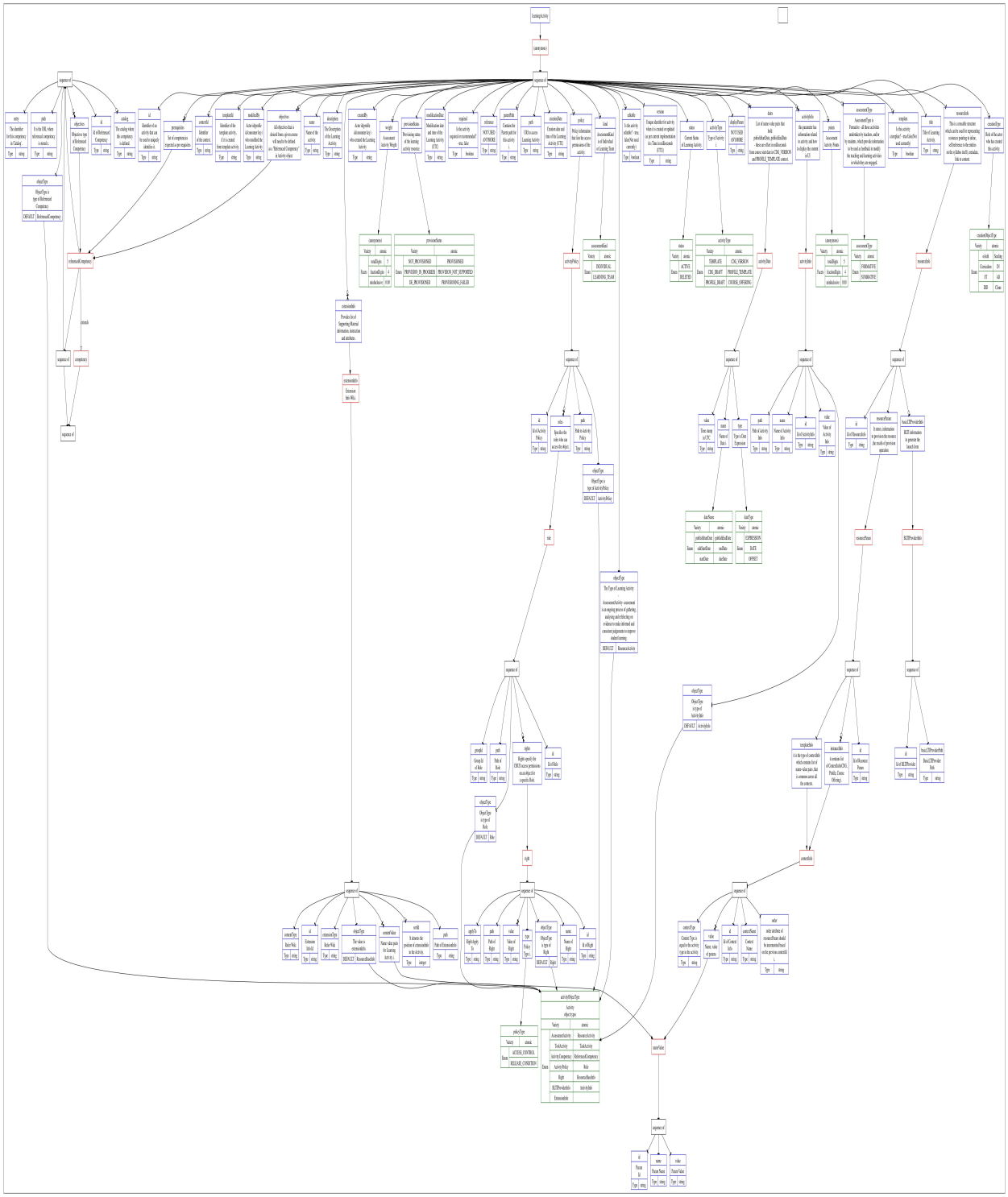


Appendix-C (Syllabus detailed model)

Tip: Zoom in to see the details!



Tip: Zoom in to see the details!



Appendix - D (Tenant integration overall system workflow)