Question 8 Use Cases

1	Would be good to replace the existing use cases where people have set up connections to oAuth at Microsoft or Google using our university accounts, and they have managed to do so without governance. This may be why we are not actually receiving a large influx of requests to support this technology, rather, just a few requests.
2	We currently use OIDC/OAuth for authentication to our primary IdM applications (administrative account management, HR enrollment of new user accounts, new user enrollments, password changes, etc.) We also use OIDC and OAuth (and -ish) via Cirrus Identity for social-to-SAML use cases. We have had a couple of instances where vendor applications could support OIDC more readily than SAML, but we haven't currently been prepared to open our existing server up broadly. (Note that in these cases, the vendors felt that it would be _easier_ to implement; they didn't already have a compatible SSO protocol. SAML, and in particular, the Shibboleth SP, was simply a difficult integration model.)
3	- Google Apps for Education - Atlassian integration
4	No specific use case at this time.
5	Our largest interest is the use of OIDC / OAuth to authorize mobile applications and APIs. We do not currently have an official OIDC / OAuth service offering, though I believe that some on our campus may be piggy-backing on our Google Apps for Education instance as an OIDC provider.
6	We would like support for end users and end user apps/applications to be able to access campus apis through an api gateway service, which allows the end user to authenticate using their familiar campus SSO (shibboleth).
7	We're a Google Apps school; as such, OIDC is important given interest around using social channels as alternative identity (i.e., Google+). In contrast, OAuth is necessary for delegated access via client software and APIs currently being deployed on campus.
8	Brown is in the process of developing and deploying a new "front door" for our alumnae. This site will be a portal (yea, that word; we're breathing new life into it). The site will obtain content from multiple backend services. Some content will come from a service providing information to the broader Brown community; other content will come from the Alumni system (essentially a CRM) and will be specific to each individual (eg giving history); I expect additional content to come from a wide variety of other systems. In the second case (CRM), it is critically important that the CRM provides the correct information -- ie the information associated with this alum, and not some other alum. Over time, the functionality in the portal will likely expand to provide Advancement staff with the ability to look at individual alumns, and groups of alumns. The security approaches in this multi-tier system should NOT offer simpler ways to impersonate an individual (especially Advancement staff) when accessing some of these backend systems. The current thinking is that the front door will be responsive, and will be programmed in javascript using node.js. This code will also run on a server (people who have disabled javascript in their browser will rely on the server based version). The code on the server will handle authentication (using passport, and strategies such as saml, LinkedIn, etc). The server will share with the javascript running in the browser a token containing information about the user; the javascript running in the browser could present this token when making requests to the various services. The front end will actually send its requests to an "API gateway" (which will be a product from MuleSoft). This approach provides a layer of abstraction, since the implementations of some of the backend systems may change over time (eg the CRM, the IDM system, etc). The API gateway will export a set of enterprise APIs to the alumnae portal, and to other systems and frontends (eg the usual apps that students want -- "today's menu"). The initial implementation will be what is described above. However, we all know that an app will have to be provided for mobile devices, and that app should support the authentication models in use on those devices.
9	The project started as a "data hub" - a central place to publish APIs that users in the community could consume to access public, and eventually, private data. To facilitate this, an API portal was built that provided OAuth support. Although we have yet to get permission from stakeholders to expose private APIs to the community, we are now using OAuth to secure access to private APIs for mobile apps that we develop in central IT. We hope to expand the use of OAuth out to the community once we get permission.
10	We are currently using OAuth for securing REST APIs. Currently, these apis are being accessed by integration ESB or by departmental IT. It doesn't use user consent at this point but it will be used in future
11	Current concern is as a consumer, rather than a provider. Have a requirement for a system that multiple parties can log into. Most of the higher education partners are likely to be eduGAIN / InCommon members, and will leverage that. However, plan to use OIDC/OAuth for authentication for institutions that aren't eduGAIN and for industry members as well.
12	Allow students' parents to "login with google" to our student information system.
13	We use cPanel for web hosting. In cPanel/WHM 11.56, functionality to support an external IdP (OIDC) was introduced, but we have no OIDC/OAuth IdP to leverage. Integrating cPanel with OIDC would eliminate redundant passwords and improve the security of the platform.
14	Some apps in the medical center already use this. Also seems to be increasingly common in securing APIs, which is our most likely use case.
15	Web apps that support it can easily integrate with our existing Google Apps domain to allow account provisioning and access through a familiar single sign-on interface.
16	We have Gluu depoyed and have bought and locally developed mobile apps using GLuu authN/authZ for login and API protection
17	Many of our members do not have SAML IDPs. For low security apps, OIDC/OAuth may be an easy way to authenticate them off of other accounts like Google Apps, Social Identities, etc

18	The main driver is Application-to-API authorization. The most common implementation of OAuth 2.0 is the so-called 3-legged flow, which expects the user to authorize access to his/her data, but student data does not belong to students, so students using a University application should not be asked whether they approve access (can you imagine asking a student if they delegate authorization in order to be invoiced for tuition?). As a result, it seems that the two-legged flow is more appropriate. But then why not use some kind of a shared-secret method? Such methods are not standard - I don't need every division creating their own implementations...and how do these get revoked when compromised? An industry-standard solution supported by central IT is a preferred option. An API, when accessed by a request with a valid token, may need the userid on whose behalf the request is being made so that authorization can be validate. For example, if a Commerce registrar is accessing the Service that provides a student's academic history, the student ID must be in the Commerce division (faculty). The API must be able to identify who the Registrar is and whether they are authorized to access the student's in the request. An OIDC JWT seems the best way to ensure that the Registrar's ID has not been modified. This means that both OAuth and OIDC need to be implemented, even for a simple two-legged flow. We are in the process of getting quotes to implement an OAuth Authorization Server, Token Service, and Policy Enforcement Point using IBM's DataPower Service Gateway (we own five of these in various environments). I still have a gap regarding OIDC. I'm investigating options.
19	We would be interested in using this to allow for our applicants, who have not yet been issued an institutional credential, to login with a credential they are more used to, as opposed to having to create another one.
20	We use the BOX API to provision box accounts for our site - this requires the use of Oath2 for the provisioning account, hence we use it for that.
21	Many vendor products purchased by campus sponsors support OAuth authentication but not SAML or OpenAM WPA authentication. It's unclear to us what additional authentication security measures are supported by these vendor products, (such as OpenID Connect), as we are still in the process of making sense of OAuth 2 vs OAuth 2 with OIDC ourselves. The more concrete use-case for us at this time is an ESB called MuleSoft, which has been purchased and is being integrated on campus. The team integrating MuleSoft on campus has requested that we enable their to authorize client API calls via OAuth token validation to our authentication service. We run OpenAM 11, which does not support OIDC natively, but does have OAuth 2 support. We are in the process of upgrading to OpenAM 13, which supports OpenID Connect out of the box.
22	We currently use OAuth/OIDC with Yahoo! And Google accounts as part of our account recovery (self-service password reset) service instead of secret questions and in combination with other information provided by account holders. Due primarily to our large scale 2-factor implementation, our account recovery service is now be analyzed for it to fit better with 2-factor. OAuth2 in combination with biometric capabilities on mobile devices is one of the potential methods that might improve account recovery functionality/usability. We are also looking at OAuth2 for web service authentication/authorization for both service accounts and end user accounts. As we continue to add vendor provided services, many private sector vendors are migrating their AD/LDAP-based offerings to either SAML2 or OAuth/OIDC (or both), but we see OAuth/OIDC providing easier to implement functionality that allows for securing service to service and service to user communication beyond just a browser-based path.
23	As faculty and staff departments look for best of class solutions for various business & teaching needs there's a growing need to integrate disparate systems in a secure way that also simplifies access to multiple resources using SSO.
24	Too early, use case development is underway and being defined
25	Mobile Forms portal in planning stages that would aggregate diverse / distributed data sources via web services which would benefit from OIDC /OAuth authentication.
26	One key use case is with the ESB that our campus is deploying. OAuth will provide improved authentication capabilities for applications making use of the ESB.
27	We need a way and instruction set to connect with our Netscaler for authentication and identity access to specific parts of our infrastructure we supply to campus users (all faculty, staff, and students). Right now I cannot find a simple way to do this, unless we use Google.edu
28	Primarily mobile usage. API use cases are interesting, but are currently system to system and are handled without end user oauth.
29	I particularly think OIDC/OAuth holds promise for providing services to alumni. It may also be promising to help current students integrate services into the single "single stream" of processing that they seem to like now.
30	InCommon is potentially interested in pursuing a pilot implementation of OpenID Connect Federation and seeing OIDC OP functionality built into popular federating software.
31	authenticating campus researchers to OIDC-enabled research services like https://docs.globus.org/api/auth/
32	External client application authentication/authorization (API consumption). Single Sign On coverage to decouple UI(s) from backend(s) via API(s) for reusability such as above.

33	Campus development efforts and third party cloud solutions supporting mobile applications, makes the need for an SSO solution critical. Consider these campus pain points that illustrate the challenges facing mobile users and organizations: SHADOW IT: The average enterprise has over 500 cloud applications in use, however fewer than 15% are enterprise ready MOBILE ACCESS: Nearly half of all cloud app activities occur on mobile devices. Yet, most mobile apps don't support SAML for SSO. For those mobile apps that do support SAML, the authentication user experience is poor and security is weakened as user sessions are not frequently revalidated. The industry is moving to solve this problem with the introduction of NAPPS or Native Applications, a standard protocol to provide SSO for users on mobile devices through a "token agent," which enables native mobile applications to authenticate users more easily. As is the case with SAML and SCIM for web applications, the promotion of NAPPS to mobile application developers is imperative to provide a more secure and integrated user experience. The NAPPS specification is part of the OpenID Foundation and is defined by the Native Applications Working Group. It is based on the OpenID Connect and OAuth 2.0 standards. It provides a seamless sign-on experience where an identity provider can federate access across numerous applications, and sessions can be validated repeatedly without degrading the user experience.
34	We needed to authenticate users through the campus Shibboleth and receive attributes via a mobile app. We used Keycloak as an identity broker to login in users through Shibboleth, and then have Keycloak manage user OAuth tokens for a mobile app. The OAuth token is then used by the mobile app to talk to an API.
35	We've had multiple people on campus requesting for delegated access to APIs by applications on behalf of, and authorized by, users. This would include things like Box API or home grown API's We've also had users wanting support for native mobile apps.
36	We need a solution for Epic and the future of API Connectivity using FHIR. If we don't have an OIDC/OAuth solution, we may be stuck with Epic being the main authorization service, which could lead to integration changes in the future.
37	We have had a few requests for OIDC/OAuth for mobile applications and for API work where a full SAML2 protocol is too much work.
38	We use OIDC in front of a number of internal applications (or student record system being the largest). We use it with a smaller number of cloud applications. We use it in front of our Moodle (LMS) instance. We use it in front of our Shibboleth server with a reverse proxy back to our IdP for all SAML applications. When we went Google 9 years ago, we made the decision to ship them our passwords - today, this makes it easier to do the above. (This has allowed faster/broader adoption of 2-step auth - since we didn't have to pay for and spend time integrating Duo.) This is our version of Identity as a Service, as was brought up in one of the I2TechEx sessions.
39	We run EPIC and various APIs and a Enterprise Service Bus. For Epic and the future of API connectivity using FHIR, if we do not have an OIDC /OAuth solution we'll likely end up with Epic being the main authorization service which could lead to integration challenges in the future.
40	As part of our healthcare division, it is required for use with the HL7 standard known as FHIR. If we do not implement an enterprise solution, our EMR system will become the defacto enterprise system which will lead to future integration challenges. We currently have an API management solution which handles the application registration process, but it is lacking a true OAuth/OIDC implementation. They do support working with plenty of 3rd party vendors overall I feel like the market is lacking with true enterprise solutions.
41	I have both use cases for native applications as well as web client applications which are using social logins (Facebook/Google/etc). Both cases would be supported much better through OIDC. I've also developed an OIDC/OAuth proxy for Shibboleth requests which I use for this purpose. I'm currently working on widening the system to work with InCommon across a variety of IdPs.
42	Three main reasons. We have implemented it for the first: 1. API securing 2. Compatibility with OIDC/OAuth IdPs 3. Native mobile support
43	There is some demand for single-page apps, SOA-based architectures, etc. plus an API manager (with low adoption rate). No demand yet for cloud apps. As I see it the real need is for JWT and OIDC/OAuth is just one way to accomplish it.
44	We already use many OAuth-protected endpoints in our custom applications. However, we are not using OIDC at this point - we have users initially authenticate via SAML and then assign OAuth tokens.
45	We plan to officially take OIDC into production for our federation (next to SAML) in the beginning of 2017. SP's can choose how they would like to offer their service through SURFconext; using SAML or OIDC. A blogpost with some more info: https://blog.surf.nl/en/connection-to-surfconext-becomes-easier-for-service-providers-with-openid-connect/
46	We need delegated access where a portal acts on behalf of the user and OIDC is easier than SAML. For details on the use case, see: https://docs.google.com/presentation/d/1BLO1_5v7Zl2CxPezICF9a71Il-fMbwCiLUDpJT3yx88/edit N.B. There are also US Relying Parties in the use case.
47	We are envisioning the creation of a portal to let (SAML) IdP administrators the possibility of linking new applications to their already existing SAML Identity Providers. For that reason, our main interest is to determine SAML<->OIDC interoperability.

48	We user the social/saml gateway now for parent access. We have three use cases we want to serve, one is to use OAuth on our mobile app. Right now we have to kludge something to make SAML work. Two, we want use google login for our applicant and alumni login. We feel this will lessen support issues. Three, we want to establish some open api's for hackathons and feel OAuth would better support that.
49	1) As an easier option for SPs to implement 2) CAS = Simple & Local, Shib = Complicated & Federated, ... 2-4 years pass ... Maybe OIDC can replace both CAS & Shib 3) Some apps only offer user-generated API keys via OIDC (some have local token mechanisms) 4) Integrate with our BuzzAPI (soa-like) API infrastructure.
50	We mainly use it with APIs for cloud hosted services.