Higher Education - Key IAM Components and Requirements

Creating Digital Identities - ID Match

Brief Description

Most universities have multiple Systems of Record through which individuals are "registered" as part of the identity management system. Typical Systems of Record include a Student System, HR system for staff, Alumni system, Guest/Affiliate System, etc. The identity match or "join" process compares identity data across Systems of Record to ensure that a single identity is created, modified, and deleted for an individual, even if that individual exists in more than one System of Record simultaneously and/or moves back and forth between Systems of Record.

Generic Functional Requirements

- Multiple source data inputs including systems of record as well as identity registry
- Data normalization
- Match on multiple attributes
- Weighted attribute and source matching
- Use business rules
- Use business process management for work flow
- Pushes unmatched identities to external manual intervention
- Database, REST, ESB, LDAP for upstream / downstream connectors

Standards Support and Integration Considerations

Key Design Considerations

- Allows Push or Pull from Systems of Record
- Allows Realtime and Batch inputs

Technical Solutions

- Open Registry
- Mural
- BPMN2

Manage Digital Identities - Self-Service

Brief Description

These are all the functions that make it easier for a person to create, update, and/or expire their digital identity. Which self-service tools a person has access to may depend on the level of assurance the system has about a person's identity and to which applications a person has access.

Generic Functional Requirements

- Ability to create one's own digitial identity (for guests)
- Ability to reset one's own passphrase (based on some stored information such as the current passphrase or questions)
- Ability to view one's own data in the system
- Ability to update one's one data (such as email address or other directory information

- Ability to send in feedback/request help
- Ability to expire (but not delete) one's own digital identity
- Ability to provide or deny access to the above based on stored information about the individual
- Logging of activity for auditing and trouble shooting purposes

Standards Support and Integration Considerations

Where possible, avoid non-standard technologies which require specifically integrated vendor components to be deployed.

Key Design Considerations

Technical Solutions

Manage Digital Identities - Administration and Delegation

Brief Description

This covers the process by which a person receives an initial digital identity or updates an existing digital identity when verification of identity cannot be handled using self-service tools. This can be due to a need for higher level of verification of identity or an inability by an individual to use the self-service tools.

Generic Functional Requirements

- Automated creation of initial digital identity based upon on-boarding data from source systems
- Automated expiration of a digital identity based upon data from source systems
- Ability to recognize and consolidate identities for an individual if the individual exists in more than one source system
- Allows for the delegation of tasks based on customer identity (such as "the students in my department")
- Allows for the delegation of tasks based on user's level of access (ie super user, assist students only, work only from a specific set of machines)
- Ability to issue a single use token for resetting of passphrase
- Ability to view/search relevant data in the system
- Ability to update data which is not based on source system data
- Ability to expire or delete a digital identity
- Ability to track authorization levels based upon level of verification of identity
- Ability to allow a 3rd party to authorize creation of a digital identity
- Logging of activity for auditing and trouble shooting purposes

Standards Support and Integration Considerations

Where possible, avoid non-standard technologies which require specifically integrated vendor components to be deployed.

Key Design Considerations

Technical Solutions

Person Data - Person Registry

Brief Description

One flexible model is to keep the IdM database of person data separate from any IdM one or more directories. The IdM's internal directory in this case ideally contains primarily authentication-related data while all authorization and related data about users and systems resides in the relational data store or registry. This IdM-private data kept in this repository is in turn replicated as may be needed to one or more external-facing read-only directories, or otherwise exposed via APIs.

Generic Functional Requirements

1. Effectively model many data relationships, including many-to-many relationships among identity data elements
2. Storage of user role data in support of IdM functions
3. Data model support for coarse-grained access control
4. Storage for data about associated or groups of application roles (related roles often provisioned in tandem)
5. Support for audit logging of activities, including support for the recording of historical changes made to sensitive data. This log would include the requester and authorizer identities, and transaction timestamps.
6. Storage of user, organization, and external application details
7. Storage for security questions and answers used for password recovery
8. Storage for data about upstream systems of record, protected applications (those under access control), and associated systems (those which require provisioning of user data)

Standards Support and Integration Considerations

Where possible, avoid non-standard technologies which require specifically integrated vendor components to be deployed.

Key Design Considerations

1. Look for designs which follow principles of master data management. For example, systems which allow to identify the authoritative source for each data item and which avoid creating unneeded replicas of such data.
2. Favor systems which easily model application-specific roles over global or institutional roles in most cases to simplify the effort needed for doing extensive roles engineering.

Technical Solutions

Systems based on commodity database solutions such as MySQL or PostgreSQL

See the IdM database design tips in the LIMA design model and the OpenRegistry community project.

Person Data - Directory Services

Brief Description

One flexible model is to keep the IdM internal person directory separate from the IdM database of identity data. The latter is primarily used in support of authorization and related functions while the directory is ideally very data lightweight and used primarily for managing authentication functions internal to the IdM web SSO and access management components. In this case, one or more externally-facing directories may be provisioned from the IdM system and exposed for such business purposes as whitepage information about people, coarse application authorization needs, etc.

Generic Functional Requirements

1. Directories can provide authentication services to primary Web SSO systems such as CAS
2. Directories can be used to manage authentication policies around passwords
3. When used for application-based authorization, the directory must store attributes about users that can provide coarse role information which applications can consume as part of their authorization logic.

Standards Support and Integration Considerations

Where possible, avoid non-standard technologies which require specifically integrated vendor components to be deployed.

Key Design Considerations

Technical Solutions

1. Commodity LDAP technologies such as:

- OpenLDAP

Authentication - Web SSO

Brief Description

Authentication is a horizontal requirement across multiple applications, platforms, and infrastructures. In general, there's no reason why user Mary should need multiple usernames. Ideally she should only need to identify herself once and then be provided with access to all authorized network resources.

The objective of SSO is to allow users access to all applications from one logon. It provides a unified mechanism to manage the authentication of users and implement business rules determining user access to applications and data.

Generic Functional Requirements

Client:

Simple client integration for multiple platforms, ex.

- Apache::AuthCAS
- Java
- JSP Client
- uPortal
- ASP.NET Forms Authentication
- ASP.NET
- ColdFusion
- Perl
- PHP
- Prado
- Python (mod_python)
- Ruby on Rails
- Seraph

- WebObjects

Server:

- Java based
- Supports application authentication with SSO server
- Support for reauth
- Support for kerberos authentication
- Support for LDAP/database connectors for identity information
- Support for 2 factor authentication

Standards Support and Integration Considerations

Key Design Considerations

Technical Solutions

- CAS Central Authentication Service <http://www.jasig.org/cas>
- SAML: Shibboleth <http://shibboleth.internet2.edu/>

Authentication - Credential

Brief Description

Authentication credential stores include Kerberos KDC, LDAP, and relational databases. Web SSO protocols can rely on credentials from any of these stores.

Generic Functional Requirements

- Support for authentication mechanisms used for Web SSO.
- Support non-web-based authentication clients
- Support for credential policies such as complexity, age requirements, etc.
- Support for throttling and locking accounts based on repeated or total bad password submissions
- Support for multi-factor credentials
- Support for replication for redundancy; master-slave or multi-master
- Support (direct or API) for self-service password change

Standards Support and Integration Considerations

Where possible, avoid non-standard technologies which require specifically integrated vendor components to be deployed.

Key Design Considerations

Technical Solutions

- web SSO technologies such as CAS or Shibboleth, integrated with credential policy controls such as those provided with OpenLDAP
- multi-factor technologies such as OTP tokens integrated with the web SSO technology

Group Management

Brief Description

Enable project managers, departments, institutions and end users to create and manage institutional and personal groups. Put control of a group in the hands of its steward and enable the person to manage the membership and what resources it can access.

Groups can be used for many purposes, such as standing committees, ad hoc research teams, departments, or classes. Key collaborative applications – mailing lists, wikis, calendars, etc. – can use this group information to make authorization decisions.

Generic Functional Requirements

- Group management from one location
- Distributed control of groups
- Integrate with student information systems and other group information source systems
- Create ad hoc groups
- Create sub groups
- Create composite groups (whose membership is determined by the union, intersection, or relative complement of two other groups)
- Support for custom group types and attributes
- Allow traceback of indirect group membership
- Delegated administration
- Allow view of individual group assignments
- Two-way integration with down stream systems

Standards Support and Integration Considerations

Key Design Considerations

Technical Solutions

The Internet2 Grouper Project <http://www.internet2.edu/grouper/>

Role Management

Brief Description

Roles and role assignment are unlikely to remain static for any length of time. Because of this, they must be managed – the entitlements associated with a role must be reviewed and updated and the users assigned the role, implicitly or explicitly, must be reviewed and changed. The business processes used to effect these reviews and changes are collectively referred to as role management (sometimes enterprise role management).

Generic Functional Requirements

Support for:

- Simple roles - sometimes called application roles
- Enterprise roles - roles that span multiple systems
- Encapsulated or layered roles - a layered role may be a business role which includes one or many application roles
- Implicit and Explicit roles
- Role mining

Standards Support and Integration Considerations

Key Design Considerations

Technical Solutions

Federation - Web Authentication

Brief Description

More and more, universities, companies, and government agencies offer services and collaborate online.
The single most important security safeguard for those tasks is the authentication of the user's identity. That is, the service provider must ensure that the person who's performing those tasks is the authorized user, not an imposter. That's where user IDs and passwords come into play as credentials to prove identity and as a prerequisite for authorized access to applications.

Within an enterprise, Single Sign-On (SSO) for all its applications in one login pass makes logistical and economic sense. When users from multiple institutions need to access these applications, the SSO of each institution can be extended to allow access to these applications outside of institutional walls in a standardized way.

A federation is a set of agreements which allow an organization to trust the authentication provided by a separate organization and provide authorization based on that authentication result. The goal of federation is to allow users to access resources in multiple organizations in a seamless manner.

Combined together the federation and SSO allow for federated SSO with local credentials that allow access to remote services.

Generic Functional Requirements

Support for:

- Federation metadata repository
- Local user/password login
- Remote login
- Database, LDAP connectors for identity information
- Individual/collective relying party configurations
- Attribute mapping
- Assertion signing
- HTTP (Post, Post-SimpleSign, Redirect), and back channel attribute release
- Identity provider attribute push/pull
- Discovery/WAYF service
- Apache and other web servers
- Java based

Standards Support and Integration Considerations

- SAML 2.0 <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security>

Key Design Considerations

Technical Solutions

- Shibboleth <http://shibboleth.internet2.edu/about.html>

Federation - Federated Provisioning

Access Management (Workflow)

Brief Description

In some cases, access to a system is automatically granted based on entitlements (if you are a student, you are automatically authorized for email) or based on a pre-defined role assignment (all employees in the Administrative Assistant I classification are authorized as purchasers in the procurement system).

Other cases require specific access request and approval workflow before an authorization is granted. Typically a user or sponsor will initiate an access request, provide some basic information, the account will be routed through an approval process, and the authorization will be provisioned to the appropriate authorization store.

Generic Functional Requirements

- Must have tools for mapping business workflow into web-based process, preferably using standard Business Process Modelling and Notation (BPMN)
- Ability to integrate with external data sources to pull information related to the request, eg display a list of academic departments to choose from which will set limits to the authorization granted (user will only be able to act on records in chosen departments)
- Base UI design should be visually appealing and intutitive
- UI elements should be easy to customize
- Must be able to accept and process access requests individually or in batches (multiple, simultaneous access requests)
- Standard process modelling should include typical players involved in access requests - user, manager, sponsor, administrative approver, etc., as well as allow for additional business functions as required for any specific integration

Standards Support and Integration Considerations

Workflow should use established, open standards than can be readily integrated with other systems.

Key Design Considerations

- How does the tool facilitate standard business process modelling, that is, intuitive screens to represent workflow which is then translated into code which supports a web-based workflow

Technical Solutions

- Bonita
- Activity
- Enhydra Shark

Audit and Reporting

Brief Description

Audit and Reporting function is used to validate that users' access to the organization's computing resources is controlled per organization's security policies.

Generic Functional Requirements

Identifying Audit worthy events, such as Add/Change/Delete to:

- Administrative activities (User accounts and Access Policies)
- User logins and Application access
- Application services availability

Method and procedure for logging events

- Logging method (SNMP, Syslog, etc.)
- Server location

Analytics for consolidated event data

- Dependent on the Regulatory Agency's requirements

Standards Support and Integration Considerations

Key Design Considerations

Event data management is easier if two key principals are followed:

- All sub-systems must follow a consistent event format
- All events must be in ASCII to improve readability

Technical Solutions

- Integrate central authorization tables and access management logs with enterprise reporting tool, like OBIEE.

Provisioning - Connectors and Adapters

Brief Description

In a provisioning system, the lack of widely adopted standards makes integration across a wide range of IDMSs and SPs difficult. The general solution has been to develop a connector framework between the engine and the IDMS and SPs. These connectors translate from the provisioning engine protocols to the target system protocols (often via REST-like APIs and execution of shell scripts).

However, each engine generally has its own connector framework. This increases lock-in, making it more difficult for institutions to leverage connectors built by others, and to switch provisioning engines when requirements change.

Generic Functional Requirements

- Must support robust and timely data synchronization with a range of external systems
- If using a vendor solution, should come delivered with a standard set of connectors for provisioning data to common target systems, such as LDAP, Active Directory, and RDMS
- Should support message queues both for notifying external systems of changes and for receiving notification of changes from external systems

Standards Support and Integration Considerations

Where possible, avoid non-standard technologies which require specifically integrated vendor components to be deployed.

Unfortunately, there are currently no standards for connector API between the provisioning engine and the connectors that are widely adapted. It may make sense to abstract out local connector integrations to allow for easier swap out of the provisioning engine should such a change become desirable.

Connectors implementing SPML are desirable, but the number of SPs supporting SPML is low.

Key Design Considerations

See the Data Integration section for general principles that apply to this section as well.

Enumerating the number and types of downstream systems to be provisioned may help determine how much any given solution will work "out of the box" and how much will need to be locally developed.

Technical Solutions

1. Most vendor IdM products come with a suite of connectors
2. See the Data Integration section for additional possible solutions

Provisioning - Engine

Brief Description

A provisioning engine is responsible for synchronizing records from an IDMS and various target service providers (SPs). This synchronization usually considers a variety of eligibility rules to determine who should have access to what services for how long and with what authorizations.

Enterprise Provisioning refers to a model of a single IDMS with a master set of identities managing SPs within the virtual boundary of the enterprise (including, say, cloud services). *Federated Provisioning* refers to a model of multiple IDMSs sharing SPs across enterprise boundaries. The requirements of enterprise and federated provisioning are similar, but not identical.

Generic Functional Requirements

- Must support robust and timely data synchronization from the source IDMS, either via notification from the IDMS or extraction of state from the IDMS
- Should support message queues both for notifying external systems of changes and for receiving notification of changes from external systems

Standards Support and Integration Considerations

Support of SPML is desirable, however the number of IDMSs, Provisioning Engines, and SPs supporting SPML is low.

There are no standards for extracting data from the IDMS, which will likely require significant custom integration effort.

Key Design Considerations

See the Data Integration section for general principles that apply to this section as well.

The design of the source IDMS will likely influence how integration with the provisioning engine is designed. IDMSs capable of supporting change notification will likely push events to the provisioning engine, while those that do not will likely require the provisioning engine to have access to the IDMS' database.

Technical Solutions

A provisioning engine can be thought of a message switch, which accepts messages or extracts information from an IDMS and sends messages to SPs via real time notification, message queues, or batch extracts.

Data Integration

Brief Description

An IdM system is designed to be an authoritative central hub of identity information. External services may access information through APIs or directory services, or data may be provisioned to external services. It is crucial to ensure that information security is maintained when data is in transport and when stored in a new location. Changes in the IdM system should be propagated to external systems in a timely manner. The ease and speed of propagating changes may be a factor when procuring systems which need to be integrated with the IdM system.

Generic Functional Requirements

1. Information about a user should include attributes as specified by the organization
2. Only the IdM system should be able to write to log/audit data stores
3. The IdM system must be able to associate user account data across multiple systems each which may have different schemes for local identifiers
4. The IdM system needs to notify downstream systems of user-related events in a timely and secure fashion
5. The IdM system must consume upstream user-related events from systems of record in a timely and secure fashion
6. IdM functions may need to be invoked by remote systems using APIs for specific purposes

Standards Support and Integration Considerations

Where possible, avoid non-standard technologies which require specifically integrated vendor components to be deployed.

Key Design Considerations

1. Look for designs of data integration components which are loosely coupled. Components which are loosely-coupled can bring flexibility and interoperability with products from different vendors.
2. Favor designs which use commodity message queuing products. For example, use products such as Apache ActiveMQ for messaging needs.
3. Integration with downstream systems ideally should be asynchronous and loosely-coupled. For example, user provisioning can use event notification mechanisms with generic user account add/modify/delete event messages.
4. Favor systems which expose IdM system functions as REST-based services for simplicity. REST-based services allow such related systems as user administration or resource management applications to simply access IdM functions.

Technical Solutions

1. Commodity messaging products such as Apache ActiveMQ
2. Integration technologies such as Apache Camel
3. REST-based web services for API exposure to external applications
4. Vendor data integration products such as Tivoli Directory Integrator