Single Sign-On and Access Management Recommended Requirements

DRAFT – September 26, 2018

Overview

This document describes a set of recommended requirements that Smarter Balanced members may voluntarily consider for inclusion in a Request for Proposals (RFP) or Statement of Work (SOW) when procuring single sign-on services. The requirements are compatible with the Smarter Balanced Single Sign-On strategy. As recommendations, they should not be taken to constrain any choices made by Smarter Balanced members.

Background

States, territories, and other members of the Smarter Balanced Assessment Consortium provide assessment-related services to educators in their districts and schools. Access to many of these services must be limited to authorized users. Services requiring access control include student registration tools, test delivery, assessment reporting, and online libraries. A Single Sign-On (SSO) and Access Management system provides central management of educator accounts and permissions across multiple applications. Centralizing access management services allows educators to use one login to access multiple services and lets states manage only one account per educator.

In many cases, SSO services are among the services that Smarter Balanced members receive under contract from their assessment service provider. However, some members contract for SSO separately from their assessment services. This arrangement enables them to use the same SSO system for all their educator services such as student information systems, learning object repositories, and so forth.

Beginning in the fall of 2018, the Smarter Balanced Assessment Consortium is planning a transition in which its Digital Library and Reporting services will subscribe to member-provided SSO solutions. This transition will grant educators the ability to use a single login to access services provided by their state and services provided by Smarter Balanced. The requirements herein are compatible with the Smarter Balanced SSO strategy, but do not substitute for the detailed integration specifications that Smarter Balanced will publish.

Authentication and Authorization

Access control involves two steps, *Authentication* and *Authorization*. Authentication is the process of determining who the user is. *Authentication* is typically done using a username and password. An SSO system lets users authenticate once and gain access to a set of related services. *Authorization* is the process of determining whether a user should have access to a particular service or feature.

Security Assertion Markup Language (SAML) is an open standard for managing authentication in web-based applications. SAML can also provide authorization services. OAuth is an open standard for managing authorization. SAML and OAuth are frequently used together. There are many open source and commercial implementations of these protocols and most SSO deployments will incorporate an existing solution.

Directory and Access Management

Single Sign-On and Access Management require a database of users. This database is called a *directory*. Typically, the directory includes the username, an encrypted password, information about the user such as ID, name, and email address, and a list of permissions granted to the user. Directories often implement the Lightweight Directory Access Protocol (LDAP) to provide access to other systems such as Access Management.

An Access Management service is an implementation of authentication and authorization protocols – typically SAML and OAuth - and it relies on a directory for its services.

Roles, Domains, and Privileges

To grant access to services, users are assigned roles. Each role assignment includes the domain of that assignment. For example, a user might be assigned, “Test Administrator for the Vista Unified School District.” In this case, the role is “Test Administrator” and the domain is “Vista Unified School District.”

A Role assignment may optionally include a subject. For example, a user might be assigned, “Test Administrator for Mathematics in the Vista Unified School District.” In this case, the role is “Test Administrator,” the subject is “Mathematics,” and the domain is “Vista Unified School District.” In this requirements set, “Subject” is described as an optional feature. Members may consider eliminating the feature, requiring it, or leaving it optional.

Privileges are particular tasks that a user is allowed to perform. Applications assign specific privileges to roles. For example, a “Test Administrator” may have the privilege to “register students for tests.” Because the mapping from roles to privileges is a function of each application, **that mapping is outside the scope of these requirements**.

Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

1 User Interface (UI)

1.1 The User Interface MUST be web-based (HTML and HTTP) and must always use encrypted communications (HTTPS).

1.2 The UI must be secured using the same Single Sign-On and Access Management services that are provided to other applications.

2 Application Program Interface (API)

(The API is an interface that allows other applications on the internet to interact with the Single Sign-On/Access Management application).

2.1 The API MUST follow RESTful design principles.

2.2 All API communications MUST be encrypted using HTTPS.

2.3 The API MUST use OAuth for authorization and security.

2.4 The API SHOULD manifest services to accomplish every function that may be performed using the UI.

2.5 The API must be secured using the same Single Sign-On and Access Management services as are provided to other applications.

3 Directory

3.1 The solution MUST implement a directory that contains an Institutional Hierarchy, User Roles, User Accounts, and Role Assignments.

3.2 The **Institutional Hierarchy** is a directory of all domains to which permissions or roles may be assigned.

3.2.1 The Institutional Hierarchy MUST include domains for the following entities: School, District, and State (or Territory).

3.2.2 The Institutional Hierarchy SHOULD support optional domains for Group of Schools and Group of Districts.

3.2.3 Each domain in the hierarchy MUST, at a minimum, have the following properties: Name, Identifier, NCES Identifier (value optional), relationship with other domains. Identifiers are typically set by the state. Regardless, they are sourced externally to the application.

3.2.4 UI and API methods must be provided for creating, updating, reading, and deleting entities in the Institutional Hierarchy.

3.2.5 The API MUST be designed in such a way to facilitate updating the Institutional Hierarchy from an external source.

3.2.6 It must be possible to set domains in the Institutional Hierarchy to an “Inactive” state so that records may be preserved for Institutions that are no longer active.

3.2.7 The UI and API MUST support queries of the domains to which a particular subdomain belongs. For example, “To which district and state does a school belong?”

3.2.8 The UI and API MUST support queries of the subdomains that belong to a particular domain. For example, “Which schools belong to a particular district?”

3.3 A **User Role** is a name for function that a user may perform such as “Test Administrator,” “Digital Library User,” or “Classroom Teacher.”

3.3.1 Each role MUST have the following properties: Name, Domains to which the role may be assigned.

3.3.2 UI and API methods MUST be provided for creating, updating, reading, and deleting roles.

3.3.3 Each role MAY have the following property: Subjects to which the role may be assigned.

3.4 **User Accounts** support authentication and provide identity information.

3.4.1 Each user account MUST have, at a minimum, the following properties: ID, email address, name (first and last).

3.4.2 UI and API methods must be provided for creating, updating, reading, and deleting User Accounts.

3.4.3 UI and API must offer methods for finding users by ID, name, and email address.

3.5 **Role Assignments** associate a role with a user and a domain.

3.5.1 Each Role Assignment MUST have the following properties: Role type (a reference to a User Role), User ID, Domain ID.

3.5.2 A Role Assignment MAY optionally have a subject property.

3.5.3 Role Assignments MUST support expiration dates. A default expiration date MUST enable the role in perpetuity.

3.5.4 UI and API methods MUST be provided for creating, updating, reading, and deleting Role Assignments.

3.5.5 UI and API methods MUST be provided for looking up role assignments by User ID and by Domain ID.

4 Access Management

The Access Management System (Identity Provider) provides single sign-on and authorization services and provides methods through which applications may gain API access.

4.1 The Access Management System SHOULD be based on existing open source Access Management software.

4.2 Access Management MUST support SAML and OAuth protocols.

4.3 The Access Management login UI should be branded to the state or client for which it is operated.

4.4 Access Management MUST offer a password reset function for those who have forgotten their passwords.

4.5 Authorization features MUST allow a client application to efficiently discover which roles are assigned to a particular user.

5 Account Administration

The account administration tool offers the UI for managing user accounts, assigning roles to users, and so forth. Administration privileges are distributed. In other words, administrators may be assigned to any domain level.

5.1 Users with Account Administrator roles MUST be able to create and manage user accounts, and assign and remove roles.

5.1.1 Account administrator privileges MUST be limited to the domain at which it is assigned to a user. For example, an account administrator at the district level may view all users who have at least one role assigned to that district or any school within that district. Likewise, an account administrator at the district level may only assign roles to domains at or within their district.

5.1.2 If a user has roles in multiple domains, account administrators MUST only be able to see the roles that are within the administrator's domain.

5.2 Administrators MUST be able to initiate password resets.

5.2.1 An administrator MUST be able to initiate a password reset for any user for which they have visibility.

5.2.2 The system MUST be secure against administrators using account hijacking to gain access to other domains.

Account hijacking occurs as follows: User A is an administrator for district X. User B has roles in both District X and District Y. User A changes user B's email address to an email box they control. Then user A initiates a password reset on user B's account. Due to their control of the email box they are able to gain control over user B's account and use that account to gain access to district Y.

The following requirements collectively prevent account hijacking.

5.2.2.1 Prohibit administrators from changing the email address for users who have one or more roles outside the administrator's domain. This prohibition MUST also apply to API access and bulk uploads.

5.2.2.2 Provide a feature whereby users may change their own email address. When notifying administrators that they cannot change an email address, include information about how a user may change their own email address.

5.2.2.3 When an administrator initiates a password reset for a user, include the name and email address of the administrator in the message to the user. (E.g. "This password reset was initiated by John Doe <john@doe.org> in their account administrator role.").

5.3 Administrators MUST be able to perform bulk import and export of data.

5.3.1 The Account Administration Administration UI MUST support bulk import and export of Domains, User Roles, User Accounts, Role Assignments, and Subjects

5.3.2 Bulk import and export MUST support CSV (according to RFC 4180). Bulk import and export MAY support Microsoft Excel (XLSX) format.

5.3.3 Import and export file formats MUST be the same for the same type.

5.3.4 Import MUST support record deletion (typically by having a column in which a DELETE keyword may be specified.

5.3.5 Bulk imports SHOULD detect the column order from the header line rather than requiring a particular column order.

5.3.6 Bulk imports SHOULD tolerate missing columns when the value is not required.

5.3.7 Bulk import and export MUST support at least 100,000 elements at a time and should facilitate convenient division into batches when more than 100,000 elements must be imported or exported.

5.3.8 Bulk import MUST detect when an uploaded element is unchanged and manage that efficiently. Multiple uploads of identical data should have no impact.

5.3.9 Bulk import MUST report on upload errors in a way that facilitates correction and subsequent retries.

5.3.10 Bulk import MUST support bulk updates of primary key information (typically email address for User Accounts and ID for all other entities).

6 OPTIONAL: Support for Limited Administrator Roles

This is an optional feature that members may choose to remove from their requirements.

A Role-Limited Administrator MUST only be able to view, grant, or revoke certain specific other roles. For example, a "Reporting Administrator" would only be able to grant or revoke reporting related roles, but not roles associated with test delivery or the digital library.

A Subject-Limited Administrator MUST only be able to view, grant, or revoke roles associated with a particular subject. For example, an "ELA Administrator" would only be able to grant or revoke roles associated with the "English Language Arts" subject.

6.1 Support the definition and assignment of role-limited administrator roles that only grant privileges to manage a specific set of other roles.

6.2 Support the definition and assignment of subject-limited administrator roles that only grant privileges to manage roles associated with a particular subject.

6.3 Support the definition and assignment of roles that are both role and subject-limited.

7 OPTIONAL: Support for Self-Registration

Self-Registration is an optional feature that members may choose to remove from their requirements.

Self-Registration allows users to register themselves and be assigned a role automatically. At Smarter Balanced, this feature is used for the Digital Library, allowing users to self-register and gain access to the Digital Library. To self-register, a user must have an email address within an authorized domain (typically assigned to a school or district) and must demonstrate control of that email address by responding to an email message.

7.1 Support configuration of self-registration for one or more specific roles. Configuration includes indicating the role to be assigned and the list of authorized email domains.

7.2 Support an integrated self-registration and password reset workflow according to the following requirements:

7.2.1 Verify password reset or self-registration through an email message that requires the user to click on a limited-lifetime link.

7.2.2 Manage users who attempt to self-register but already have an account by offering to reset their password. For security reasons, the disclosure that they already have an account must be done by email. Otherwise, anyone could discover whether accounts exist.

7.2.3 Manage users who attempt a password reset but don't have an account by offering to self-register. This process should only be done for users with emails that belong to authorized domains. For security reasons, the offer to self-register should be made by email.

7.2.4 Prohibit users who self-register from changing their email address to a domain that is not authorized for self-registration. This precaution is to prevent user from repeatedly self-registering and then giving the account away to another by changing their email address.

8 **OPTIONAL: Support for User Activity Reporting**

User Activity Reporting is an optional feature that members may choose to remove from their requirements.

User Activity Reporting keeps a record of user logins and reports on utilization of services by client.

8.1 Maintain a record of each login and the service that requested the login (e.g. Reporting, test administration, digital library, etc.)

8.2 Reports MUST list the count of registered and active users. Active users are those that have logged in within a certain period of time (e.g. 30 days or 90 days). The period should be configurable as a global value (all activity reports use the same period to define “active users”).

8.3 Users with administrative privileges MUST be able to generate reports within their domain. For example, a district administrator should be able to generate a report for their district with breakdown by schools. A state administrator should be able to generate reports for the state with breakdown by districts and schools.

8.3.1 Activity reports MUST support filtering by subscribing application. Example applications: Digital Library, Reporting, and Test Delivery.

8.3.2 All reports MUST be viewable in the web browser and exportable to .CSV format.

9 **OPTIONAL: User Groups**

User Groups is an optional feature that members may choose to remove from their requirements.

A group is a set of users that are assigned a common set of roles. The group is associated with an entity (State, District, Institution, etc.). Institutional association is strictly to control which administrators can manage the group.

9.1.1 The Account Administration Tool MUST provide a UI for creating and managing groups and the associated role assignments. The group view should show and manage all users who are members of the group. The user view should show and manage all groups in which the user is a member.

9.1.2 Bulk import and export MUST support import and export of group membership.

9.1.3 The API MUST include group maintenance functions.

10 Audit Trail

10.1 Maintain an audit trail of user creation, update, and delete operations. Since the email address is used as the primary key, email address changes MAY require special attention in the audit trail.

10.2 Maintain an audit trail of all role grants and revocations.

10.3 Maintain an audit trail of group membership changes.

10.4 Audit trail records MUST include when the operation was performed, by which account, and through which entry point (UI, Bulk Upload, API).

10.5 Record the audit trail in a standard format such as Syslog (RFC 5424) suitable for analysis by popular log and audit tools.

11 Platform

11.1 Application SHOULD be developed in a common language and platform such as Java, JavaScript (Node.js), PHP, etc.

11.2 Application SHOULD be containerized using Docker or a similar technology suitable for easy deployment in a cloud-hosted infrastructure.

12 Capacity and Performance Requirements

12.1 Solution MUST support more than 2 million total registered users, 500,000 users concurrently logged into services, and 5,000 administrators concurrently using the Account Administration tool.

12.2 Response time to successful login operations MUST be less than 500ms.

12.3 Response time to administration tasks MUST be less than 1 second for most operations and less than 10 seconds for search operations.

13 **Additional Requirements**

13.1 Accessibility Standards: All user Interfaces must meet WCAG 2.0 AA Accessibility requirements

13.2 Mobile Device Support: All user-related features, including login, self-registration, and password reset, must be convenient to use from a mobile device such as a phone or tablet.

13.3 Browser Support: System MUST support all contemporary browsers in broad use, including current versions of Microsoft Edge, Google Chrome, Apple Safari, and Mozilla FireFox.

14 Documentation

14.1 Write a User Guide documenting the login interface and the Account Administration tool.

14.2 Write an Operations Guide documenting how to configure the full system and how to configure applications subscribing to the SSO and Access Management Solution.

14.3 Write a Developers Guide documenting how to integrate subscribing applications. The guide SHOULD reference SAML and OAuth standards and offer sample code for primary operations.

14.4 Write a Deployment and Maintenance Guide indicating how to deploy the application, configure it, and test the solution to ensure it is functioning properly.

14.5 Write a Software Maintenance Guide describing the architecture of the application, indicating how to set up a development environment and how to build and distribute updates to the application.

14.6 Administer a document review process to ensure that all documents meet client requirements.

15 Testing

15.1 Perform functional testing of all components and operations. Develop automated unit tests for key functions.

15.2 Run performance tests to ensure that the solution meets capacity requirements.

15.3 Develop a comprehensive regression test perform testing before the User Acceptance Test.

15.4 Administer a User Acceptance Test in coordination with client staff.

Project Deliverables

Supplier will provide and maintain a project schedule including key milestones and all deliverables. Any change in the schedule must be approved by client.

Deliverables shall include:

1. Functional requirements document.
2. Design documents and associated technical specifications.
3. All software developed or enhanced as part of this project including, but not limited to: software product, development tools, support tools, data migration software, integration software, and installation software.
4. Test plan for and results of unit testing, functional testing, performance testing, and user acceptance testing.
5. Deployment and validation scripts, artifacts, and software containers.
6. Documentation including user guides, operations and maintenance guides, deployment guides, and development platform configuration.
7. Deployment of final code to staging and production servers.

All software, including enhancements, new code, deployment scripts, test scripts and so forth must be stored in a source code repository designated by client (e.g. GitHub). Supplier is strongly encouraged to follow best practices for Open Source Development including using the source code repository as the primary code repository for the development effort. All documentation, including user guides, deployment instructions, and so forth, must be included in the source code repository.

Intellectual Property Terms

The following is are suggested intellectual property terms based on those used by Smarter Balanced. Member should substitute their standard IP terms or modify these to meet their needs.

All Deliverables are considered work for hire and become the property of the client with the expectation that software and documentation will be made publicly available under open licenses. Client prefers the Educational Community License 2.0 (ECL 2.0). Existing source code and enhancements to that code may be bound by existing open source licenses.

Questions for Potential Providers

The following questions are a starting point for evaluating supplier proposals. Depending on your procurement process you may consider including them in an RFP, using them in the proposal evaluation process, or both.

- Please describe your experience with single sign-on and access management solutions. Which products and protocols have you used? What unique qualifications do you have for this project?

- What technology and platform do you propose to use for Single Sign-On and Access Management services? Why did you choose this particular solution?

- Which database technology do you propose to use for the directory information, including Institutional Hierarchy, User Roles, User Accounts, and Role Assignments? What are the advantages of your choice?

- Will directory information be retained in a single database or will it be replicated between multiple services? If replicated, what technology do you propose to use to maintain integrity between the copies? What are the advantages and disadvantages of the design you chose?