High Level Design

Smarter Balanced Assessment Consortium Test Delivery System

Components: Permissions

Revision History

|  |  |  |
| --- | --- | --- |
| **Revision Description** |  | **Date** |
| First Release | David Lopez de Quintana | 30 August 2013 |
| Updated mockups and API signatures | David Lopez de Quintana | 09 September 2013 |
| Updated APIs to get al related data with every API call | David Lopez de Quintana | 10 September 2013 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

­­

Contents

[1 Purpose 3](#_Toc366571379)

[2 High Level Design 3](#_Toc366571380)

[2.1 Background Information 3](#_Toc366571381)

[2.2 High Level Design 3](#_Toc366571382)

[2.3 Key Functions 4](#_Toc366571383)

[2.4 Deployment 4](#_Toc366571384)

[2.5 Managed Data 4](#_Toc366571385)

[2.6 REST Interfaces 5](#_Toc366571386)

[3 User Interface 6](#_Toc366571387)

Figures

[Figure 1. Permissions and Related Components 3](#_Toc366571388)

[Figure 2. Home Screen Mockup 7](#_Toc366571389)

[Figure 3. View/Edit Roles Mockup 7](#_Toc366571390)

[Figure 4. View/Edit Components Mockups 8](#_Toc366571391)

[Figure 5. View/Edit Permissions Mappings 8](#_Toc366571392)

[Figure 6. View/Edit Mappings 9](#_Toc366571393)

Tables

[Table 1. Public REST APIs 6](#_Toc366571394)

[Table 2. User Interface Screens 6](#_Toc366571395)

# Purpose

The purpose of this document is to provide a high level design and technical direction on the development of the Permissions component.

# High Level Design

## Background Information

The Architecture Report describes the Test Registration component as follows:

*This is a centralized permissions management for the systems components. It is necessary to require that components share the same permissioning capabilities in order to reduce permissions management complexity, and to allow consistent user experience across multiple components developed by different vendors.*

Permissions is one of the Shared Services as implied by the Architecture Report. Its primary functions are as follows:

* Master repository of role data: Permissions stores user roles and is the authoritative source for role information
* Master repository of permissions: in this context, a “permission” refers a function or unit of functionality that a components offers to a user, and to which a user either does or does not have permission to access.

## High Level Design

The following diagram illustrates the high level design of the Permissions component and how it fits in with components that it is responsible for communicating with.



Figure 1. Permissions and Related Components

The following corresponds to the numbers in this diagram.

1. Administrative users use the Permissions user interface to create/manage roles, create/manage permissions and create/manage mappings between them.
2. Client components use REST APIs provided by permissions to determine which roles have mappings to the component’s specific permissions, and to determine what permissions each role allows.
3. Test Registration also uses the same REST APIs as above to obtain role information as a client component like any other, but is also interested in the allowed hierarchy level of the entity that can be associated with a given role. It users this information to validate user files and to present in its user interface.
4. User files provided to Test Registration are validated against the information provided by Permissions. Specifically.
   1. Roles provided in user records must exist in Permissions
   2. Entities associated with roles must be at a level consistent with the allowable level information provided by Permissions
5. Permissions provides user information to SSO to CRUD user accounts.

## Key Functions

## Deployment

Permissions is a member of the Shared Services component family. Permissions must be deployed with every deployment level as all components at a given deployment level rely on it.

## Managed Data

The data managed by Permissions is described as a DTD and an example XML. This format is selected as a means of formally specifying what data is required, but we are not requiring that the data be stored, extracted or managed in XML format. Other formats such as JSON are acceptable as long as they meet the specification of what data is managed.

The highest level specified by the DTD are Permissions, and contain one or more roles, components and permission mappings.

<!ELEMENT permissions (role+, component+, permissionmapping\*)>

Each role has a name and contains one or more allowable entity types. The allowable entity type indicates what types of entities can be associated with a given role.

<!ELEMENT role (allowableentitytype+) >

<!ATTLIST role

name CDATA #REQUIRED

>

<!ELEMENT allowableentitytype EMPTY >

<!ATTLIST allowableentitytype

entity ( CLIENT | GROUPOFSTATES | STATE | GROUPOFDISTRICTS | DISTRICT | GROUPOFINSTITUTIONS | INSTITUTION ) #REQUIRED

>

Components contain permissions which are functions or groups of functionality that a component wishes to control based on role. A user is either granted or denied access to functionality (represented by a permission) based on role.

<!ELEMENT component (permission+) >

<!ATTLIST component

name CDATA #REQUIRED

>

<!ELEMENT permission EMPTY >

<!ATTLIST permission

name CDATA #REQUIRED

>

Finally, roles and permissions are associated by means of a permission mapping.

<!ELEMENT permissionmapping EMPTY >

<!ATTLIST permissionmapping

role CDATA #REQUIRED

permission CDATA #REQUIRED

>

The following example XML illustrates the usage pattern.

<?xml version="1.0" encoding="utf-8"?>

<permissions>

<role name="Test Author">

<allowableentitytype entity="CLIENT"/>

<allowableentitytype entity="STATE"/>

</role>

<role name="Test Approver">

<allowableentitytype entity="CLIENT"/>

<allowableentitytype entity="STATE"/>

</role>

<role name="Test Kingpin">

<allowableentitytype entity="CLIENT"/>

<allowableentitytype entity="STATE"/>

</role>

<component name="TestAuthoring">

<permission name="TA|Author Tests"/>

<permission name="TA|Approve Tests"/>

<permission name="TA|Release Tests"/>

</component>

<permissionmapping role="Test Author" permission="Author Test"/>

<permissionmapping role="Test Approver" permission="Author Test"/>

<permissionmapping role="Test Approver" permission="Approve Tests"/>

<permissionmapping role="Test Kingpin" permission="Author Test"/>

<permissionmapping role="Test Kingpin" permission="Approve Tests"/>

<permissionmapping role="Test Kingpin" permission="Release Tests"/>

</permissions>

## REST Interfaces

The following public APIs are required for components to access role and permission data. Permission mappings are not explicitly exposed but are used to obtain roles with mapping to components and permissions, components with mappings to roles and permissions with mappings to roles. The public APIs are read-only as client components are not allowed to change role and permission data.

|  | **Description** | **Answers this Question** | **APIHandler** |
| --- | --- | --- | --- |
| api/role | Get all roles including allowable entities | What are all the roles including allowable entities? | getRoles |
| api/role?role={roleName} | Gets a specific role including allowable entities | What are the allowable entities for this role? | getRoles |
| api/role?component={componentName} | Gets all roles (including allowable entities) that have mappings to a specific component | What are all the roles that have mappings to this component? | getRoles |
| api/role?component={componentName}&  permission={permissionName} | Gets all roles (including allowable entities) that have mappings to a specific component and permission | What are all the roles that have mappings to this component and this component’s permission? | getRoles |
| api/component | Gets all components including permissions | What are all the components including their permissions? | getComponents |
| api/component?component=  {componentName} | Gets a specific component including permissions | What are the permissions for this component? | getCompononts |
| api/component?role={roleName} | Gets all components (including permissions) that have mappings to a specific role | What are all the components that have a mapping to this role? | getComponents |
| api/permission?role={roleName}& component={componentName} | Gets all permissions that have mappings to a given role for a given component | What are all the permissions for this component that have mappings to this role? | getPermissionByRole-AndComponent |
| api/deleteRole?roleId={roleName} | Deletes a role by role name. | Is a particular role deleted successfully? | deleteRole |
| api/entity | Gets all AllowableEntityTypes | What are all AllowableEntityTypes? | getAllAllowableEntities |

Table 1. Public REST APIs

Private REST interfaces that mutate data to support the user interface may be established at the discretion of the detailed designer.

# User Interface

The following table indicates the screens that are required for the Permission application.

| **Function** | **Description** |
| --- | --- |
| Home | This screen provides the main navigation |
| View/Edit Roles | This screen provides the means to add, edit or delete roles |
| View/Edit Components | This screen provides the means to add, edit or delete components |
| View/Edit Permissions | This screen provides the means to add, edit or delete permission |
| View/Edit Mappinngs | This screen provides the means to create or delete mappings between roles and permissions |

Table 2. User Interface Screens

The following mockups describe the user interface of Permissions.

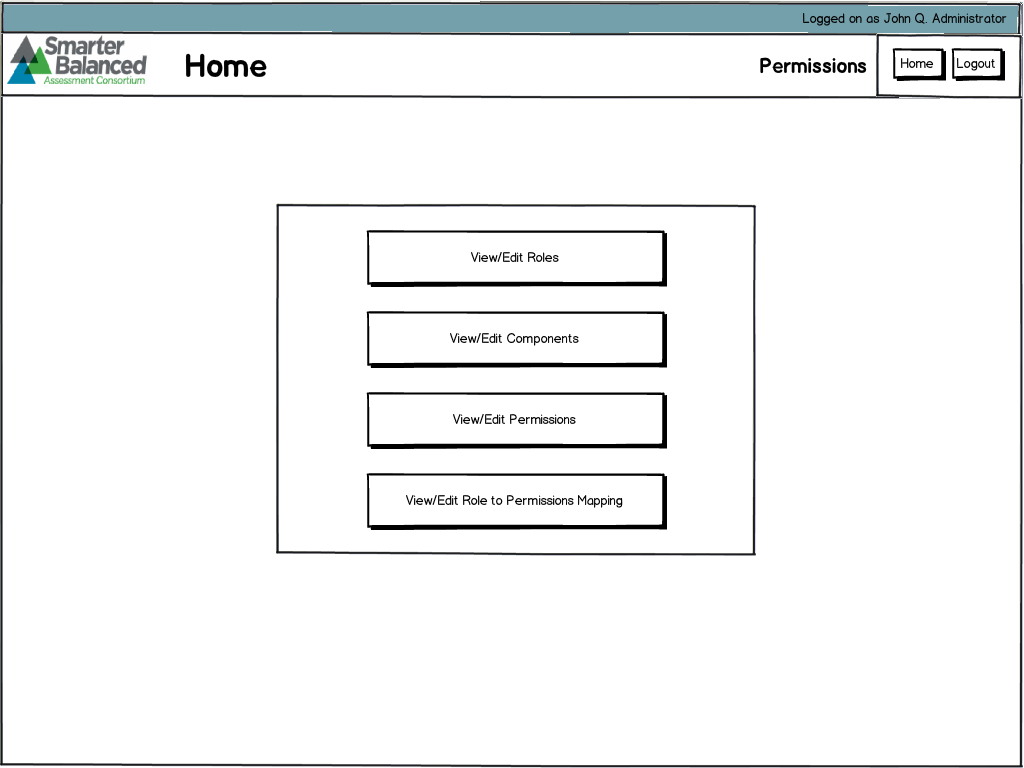


Figure 2. Home Screen Mockup

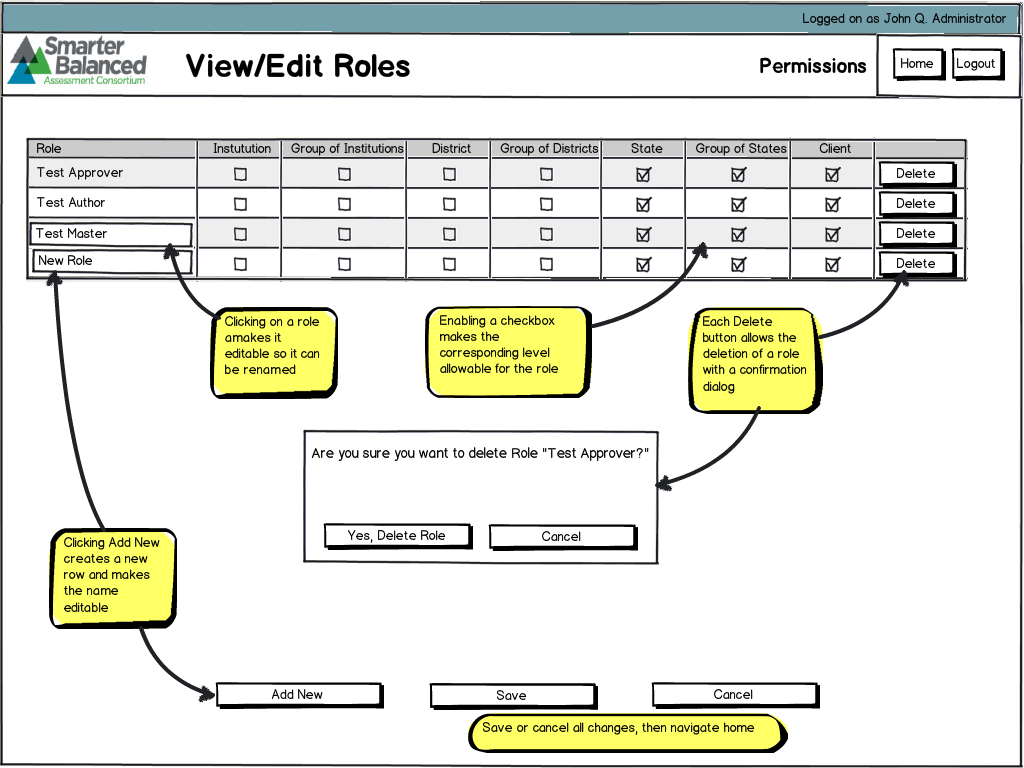


Figure 3. View/Edit Roles Mockup

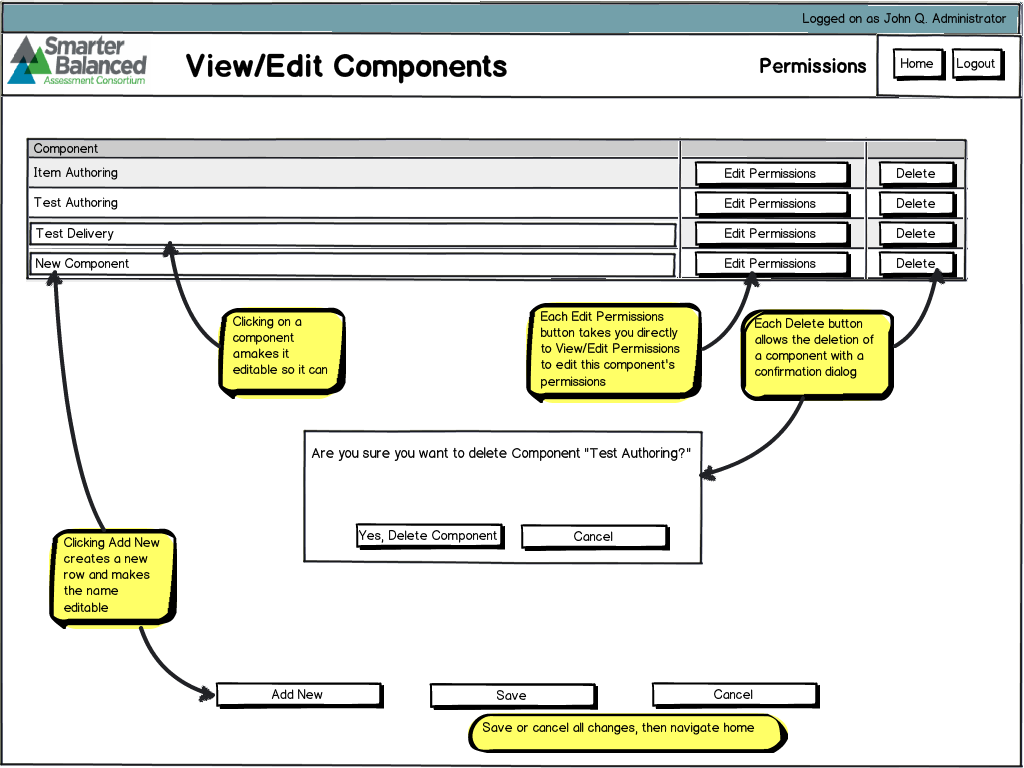


Figure 4. View/Edit Components Mockups

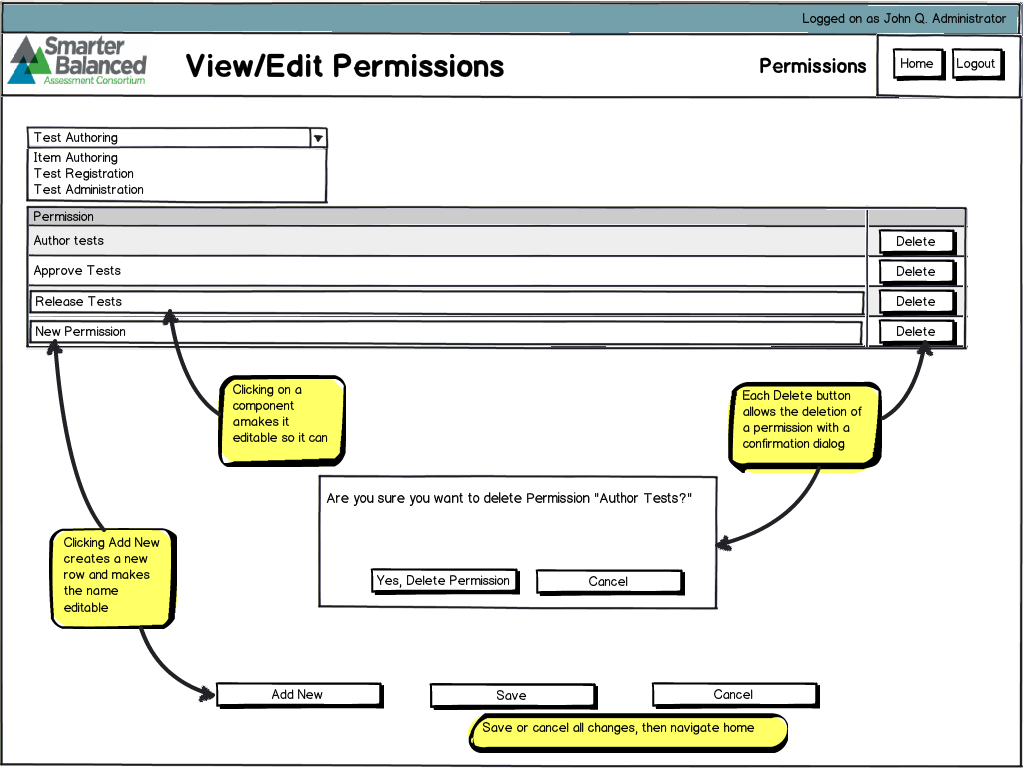


Figure 5. View/Edit Permissions Mappings

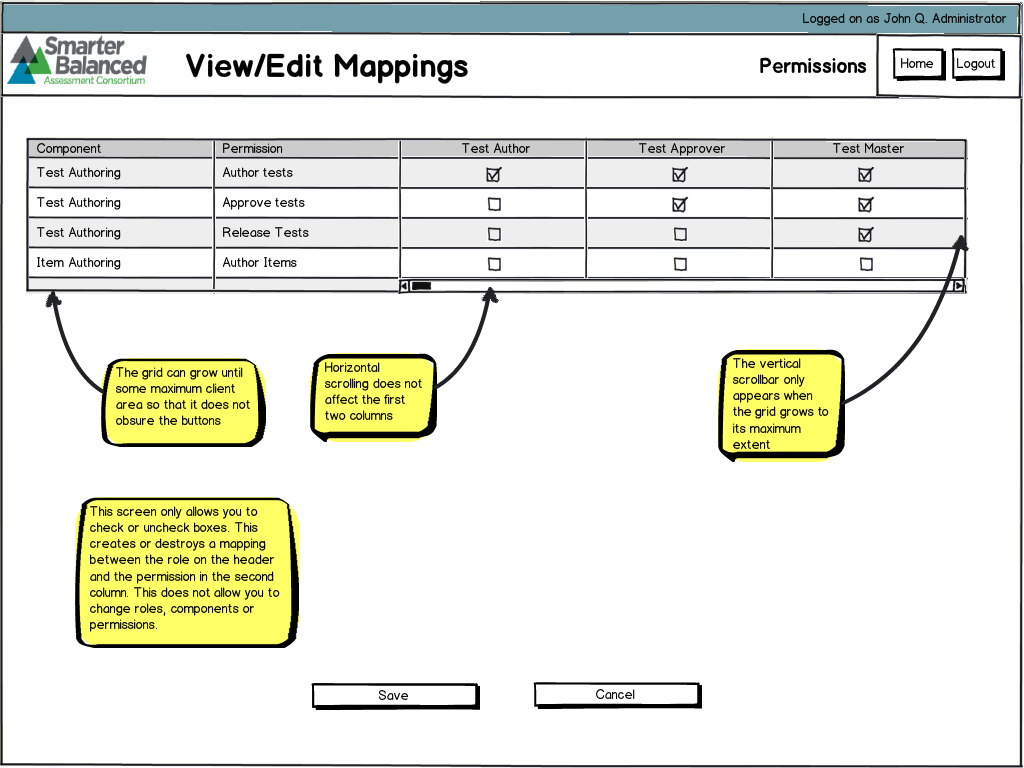


Figure 6. View/Edit Mappings