Mobile Application Store Authentication Service Requirements

Author: Eric Gieseke

Date: 10/31/2013

Introduction

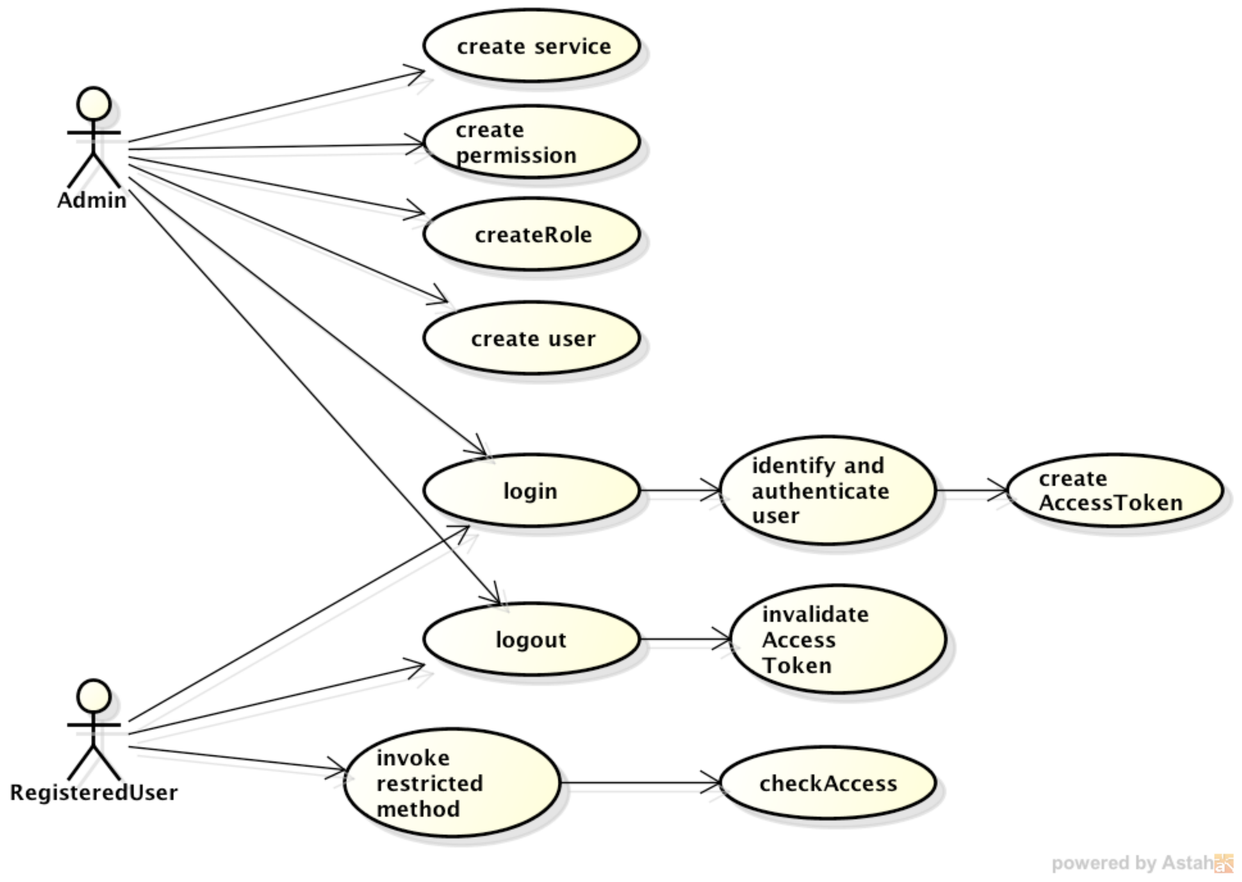
This document provides the requirements for the Mobile Application Store Authentication Service.

Overview

Mobile Application stores provide a source of applications, ringtones, and wallpapers for mobile consumers to download to and access from their mobile devices. A wide variety of applications are available, including games, office applications, and social applications. Ringtones and wallpapers allow users to customize their mobile experience. Both free and premium (e.g. paid) content is available for download.

Authentication Service

The Authentication Service is responsible for controlling access to the Mobile Application Store restricted interfaces. The Authentication Service provides a central point for managing Users, Services, Permissions, Roles, and Access Tokens.



The Authentication Service supports 2 primary actors or roles: The Administrator and Registered Users. The Administrator is responsible for managing the Services, Permissions, Roles, and Users maintained by the Authentication Service. The Registered User is the consumer of the the Authentication Service, using it to login, logout, and indirectly to access restricted service methods.

Authentication Service API

The Authentication Service API is needed to support the following functions:

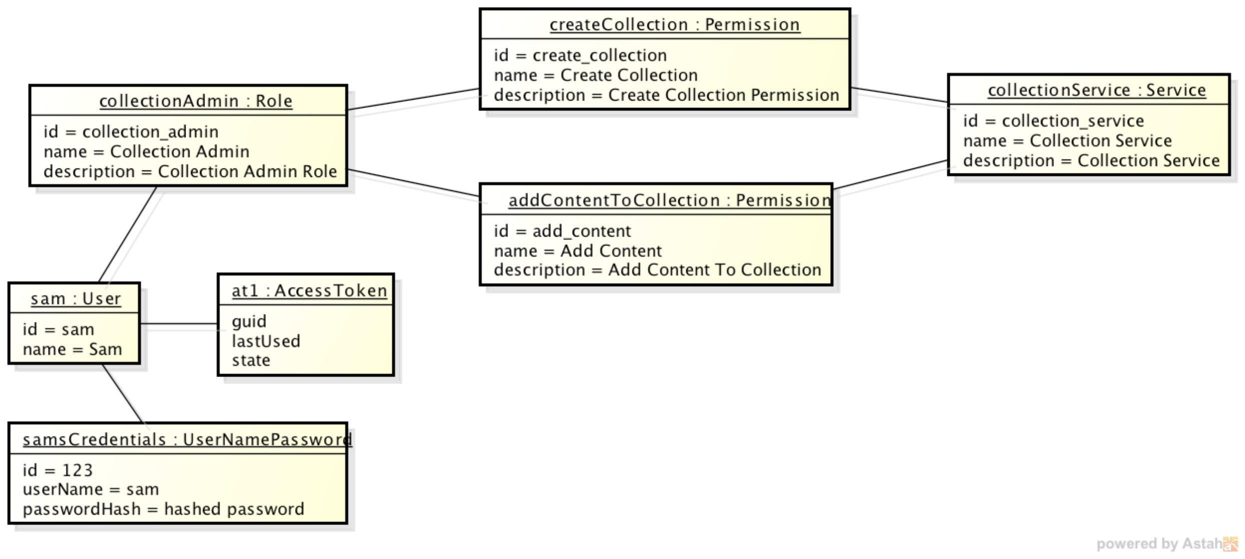
- Creating Services:
    - Service aggregates a set of permissions specific to the Service

  - A Service has a unique ID, a name, and a description.
  - A Service represents any of the Mobile Application Store services, including the Product API, the Collection Service, and the Authentication Service.
  - Restricted access
- Creating Permissions:
  - Permissions represent an entitlement required to access restricted functionality of the Mobile application store. Restricted service methods require that the user accessing the method have the permission associated with the method.
  - Permissions are specific to a Service.
  - Permissions have a unique id, name, and description.
  - A Service may have one or more permissions.
  - A User may be associated with zero or more permissions.
  - Restricted access
- Creating Roles:
  - Roles are composites of Permissions.
  - Roles provide a way to group Permissions and other Roles.
  - Like Permissions, Roles have a unique id, name, and description.
  - Users may be associated with Roles, where the user has all permissions included in the Role or sub Roles.
  - Roles help simplify the administration of Users by providing reusable and logical groupings of Permissions and Roles.
  - Restricted Access
- Creating Users:
  - Users represent registers users of the Mobile Application Store.
  - Users have an id, a name and a set of Credentials. Credentials include a username and a password. To help protect the password, the password should be hashed.
  - Users are associated with 0 or more Roles or Permissions.
  - Restricted Access
- Login:
  - The Login process provides users AccessTokens that can then be used to access restricted Service Methods.
  - Login accepts a User's credentials (username, password).
  - A check is made to make sure that the username exists, and then that the hash of the password matches the known hashed password.
  - If authentication fails, an AuthenticationException should be thrown.
  - If authentication succeeds, an AccessToken is created and returned to the caller.
  - The accessToken binds the User to a set of permissions that can be used to grant or deny access to restricted methods.
  - AccessTokens can timeout with inactivity.
  - AccessTokens have a unique id, an expiration time, and a state (active or expired).
  - Access tokens are associated with a User and a set of Permissions.

- Logout:
    - Logout marks the given Access Token as invalid.
    - Subsequent attempts to use the AccessToken should result in a InvalidAccessTokenException.
- Invoking Restricted Methods:
    - All restricted methods defined within the Mobile Application Store should accept a AccessToken
    - Each Restricted method should validate that the AccessToken is non null and non empty.
    - The Restricted method should pass the accessToken to the Authentication Service with the permission required for the method.
    - The AuthenticationService should check to make sure that the AccessToken is active, and within the expiration period, and then check that the user associated with the AccessToken has the permission required by the method.
    - The AuthenticationServce should through a AccessDeniedException or InvalidAccessTokenException if any of the checks fail.

Exceptions should include useful information to help users understand the nature of the Exception.

The following instance diagram shows how the User, Role, Permission, Service, AccessToken, and User Credentials are related.


createCollection : Permission
id = create_collection
name = Create Collection
description = Create Collection Permission
collectionAdmin : Role
id = collection_admin
name = Collection Admin
description = Collection Admin Role
collectionService : Service
id = collection_service
name = Collection Service
description = Collection Service
addContentToCollection : Permission
id = add_content
name = Add Content
description = Add Content To Collection
sam : User
id = sam
name = Sam
at1 : AccessToken
guid
lastUsed
state
samsCredentials : UserNamePassword
id = 123
userName = sam
passwordHash = hashed password
powered by Astah

Restricted methods include restricted methods on the AuthenticationService, Collection Service, and Product API Service.

Sample Authentication Data:

# define_service, <service_id>, <service_name>, <service_description>

define_service, product_api_service, Product API Service, Product Management and Access

# define service

# define_service, <service_id>, <service_name>, <service_description>

define_service, product_api_service, Product API Service, Product Management and Access

define_service, collection_service, Collection Service, Collection Management and Access

define_service, authentication_service, Authentication Service, Manage Authentication Configuration and Control Access to Restricted Service Interfaces

# define permissions

# define_permission, <service_id>, <permission_id>, <permission_name>, <permission_description>

define_permission, collection_service, create_collection, Create Collection Permission, Permission to create a new collection

define_permission, collection_service, add_content, Add Collection Content Permission, Permission to create a new collection

# define roles

# define_role, <role_id>, <role_name>, <role_description>

define_role, collection_admin, Collection Admin, All permissions required by collection administrators

# add entitlement (permission or role) to role

# add_entitlement_to_role, <role_id>, <entitlement_id>

```
add_entitlement_to_role, collection_admin, create_collection
add_entitlement_to_role, collection_admin, add_content

# create_user
# create_user <user_id>, <user_name>
create_user, sam, Sam
# add_credential
# add_credential <user_id>, <login_name>, <password>
add_credential sam, sam, secret
```