Smart City Authenticator Design Document v1

### Date: 16 Nov 2020

Author: Andrew Pham

Reviewer(s): Ky Nguyen, Behrooz Amirzadeh

Introduction

This document defines the design for the Smart City Authenticator

# Overview

The authentication service allows the controller service access to restricted methods when controlling the model service. Only authorized users are allowed certain entitlements. The controller asks for authentication and if the user has the correct entitlements and credentials the authentication service will pass the controller an authentication token allowing access.

# Document Organization

1. Requirements
2. Use Cases
3. Implementation
4. Sequence Diagram
5. Class Diagram
6. Class Dictionary
7. Interfaces
8. Exception handling
9. Testing
10. Risks

# I. Requirements

This section provides a summary of the requirements for the Smart City Controller.

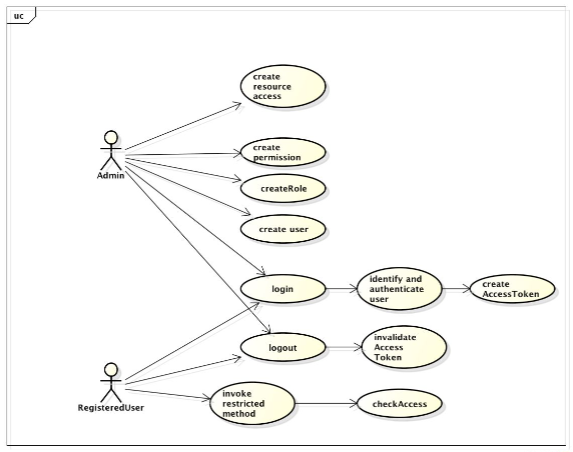
1. Allow the user to define entitlements and authentication structure with roles, permissions, and resources
2. Restrict methods to require an authentication token
3. Update the model service and command service to use the authentication service
4. Create a method of login and verification for script commands and voice commands
5. Authenticate microphone voice commands via biometrics on behalf of the person making the request, not the person executing the script

II. Use Cases

This design supports the following use cases:

Administrator: Create a structure for the authentication service by defining and composing users, roles, permissions, and resources.

Registered User: Access restricted methods by logging in and obtaining an authentication token.

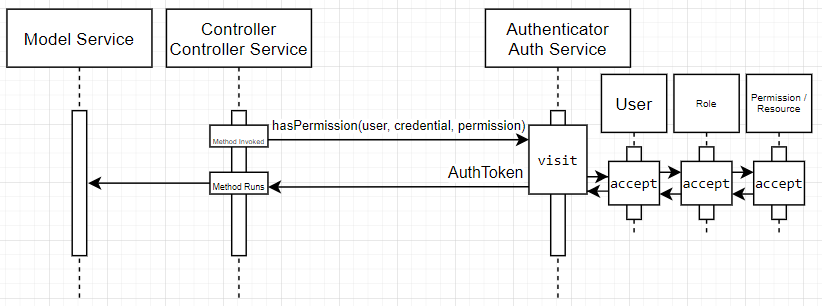


Use case diagram

III. Implementation

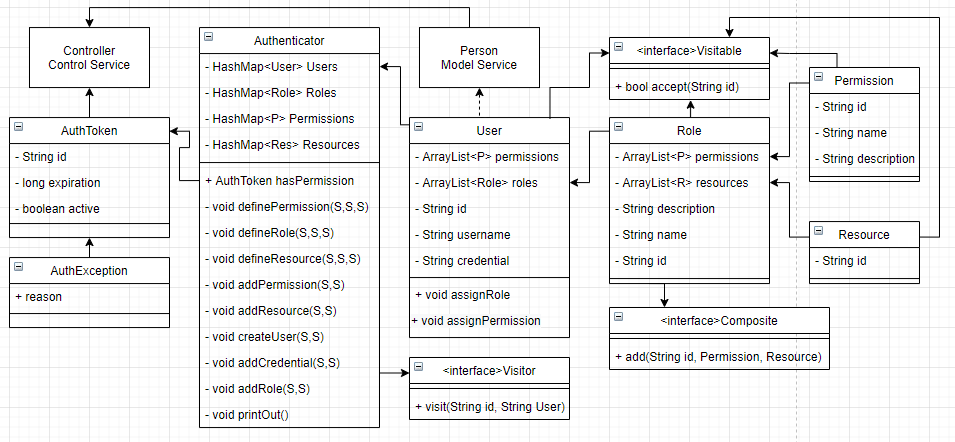
### Users contain roles which defines their permissions and resources. When a method is invoked from the script the controller will check to see if the logged in user has permission by calling hasPermission on the authentication service which visits all entitlements assigned to the user and checks the credentials. If the user has the permission / resource, the user will return an authentication token allowing the controller to perform the method. If a method is invoked via microphone sensor event the controller searches for the subject’s biometric and passes it to the authenticator which creates a temporary session to invoke the voice command on the speaker’s behalf.

IV. Sequence Diagram



A sequence diagram showing the controller invoking a restricted method and calling has permission to check for authentication. The authenticator visits entitlements to check for permission, then sends back an authentication token so the controller can run the method and alter the model.

V. Class Diagram



A class diagram of the Authentication Service.

VI. Class Dictionary

## Authenticator

The Authenticator issues authentication tokens and keeps track of users and defined entitlements. The authenticator is a singleton and a visitor. The visitor pattern allows the authenticator to visit roles, permissions, resources to check for access. The command module calls specific authenticator methods to create the authentication system. If the command module wants to invoke a restricted method it calls hasPermission to check for authorization.

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| users | HashMap<User> | List of defined users |
| roles | HashMap<Role> | List of defined roles |
| permissions | HashMap<Permission> | List of defined permissions |
| resources | HashMap<Resource> | List of defined resources |

**Methods**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| hasPermission | Param: String user, String credential, String permission  Return: AuthToken | Checks if user has permission and if so returns AuthToken. If not, returns blank AuthToken |
| definePermission | Param: String id, String name, String desc  Return: void | Defines a permission |
| defineRole | Param: String id, String name, String desc  Return: void | Defines a role |
| defineResource | Param: String id  Return: void | Defines a resource |
| addPermission | Param: String role, String permission  Return: void | Adds permission to role |
| addResource | Param: String role, String permission  Return: void | Adds resource to role |
| createUser | Param: String id, String username  Return: void | Defines a user |
| addCredential | Param: String id, String credential  Return: void | Assigns a credential to a user |
| addRole | Param: String id, String role  Return: void | Assigns a role to a user |
| printOut |  | Prints authentication service overview to console |
| visit | Param: String id, String user  Return: boolean | Visits user’s entitlements to check for authorization |

## User

A user represents a person in the model. Users are associated with entitlements.

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | String | Unique identifier |
| username | String | Login username |
| credential | String | A password or biometric |

**Methods**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| assignRole | Param: String id, String role  Return: void | Assigns a role to the user |
| assignPermission | Param: String id, String permission  Return: void | Assigns a permission to the user |
| accept | Param: String id  Return: boolean | Accepts a visitor and checks if user has authorization by visiting its entitlements |

**Associations**

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| roles | HashMap<Role> | List of roles assigned to the user |
| permissions | HashMap<Permission> | List of individual permissions assigned to the user |

## AuthToken

The Authentication Token provides access to restricted methods in the controller. A user verifies their permission and if the verification succeeds the token is sent to the controller giving access to the method. AuthTokens time out.

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | String | ID for this specific authtoken |
| expiration | long | Time of day in milliseconds plus two minutes |
| active | boolean | True if entitlements found and credentials match |

## Role

A Role is a composite of permissions and/or resources. A role with access to a resource is a resource roll.

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | String | Unique identifier |
| description | String | Description of role |
| name | String | Name of role |

**Methods**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| accept | Param: String id  Return: boolean | Accepts visitor and checks if role has the entitlement by visiting role’s permissions and resources |

**Associations**

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| resources | HashMap<Resources> | List of resources assigned to the user |
| permissions | HashMap<Permission> | List of individual permissions assigned to the role |

## Permission

Permissions define a user’s or role’s entitlements. Permissions are the leaves of role composites.

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | String | Unique identifier |
| description | String | Description of permission |
| name | String | Name of permission |

## Resource

A resource describes a logical entity attached to a role. A resource role allows access to the resource. Resources are leaves of role composites.

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| id | String | ID of resource |

VII. Interfaces

## Visitor

Visitor is part of the visitor design pattern. Visitor checks multiple types of entitlements for authentication. The Authenticator is a visitor.

**Methods**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| visit | Param: String id, String User  Return: boolean | Returns true if visitable has the entitlement |

## Visitable

Visitable is part of the visitor design pattern. It is used so that the authenticator can visit different parts of the composite to check for authentication.

**Methods**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| accept | Param: String id  Return: bool | Accepting a visitor checks credentials either login, faceprint, or voiceprint. Returns true if match. |

## Composite

Composite is part of the composite design pattern. It allows us to create a hierarchy of objects that compose another object. In our case we have a composite of permissions and resources in roles.

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| add | String id, Permission P, Resource R | Adds permissions or resources to the composite |

VIII. Exception Handling

An AuthException will be thrown if the user does not have the entitlements to perform the command or if the credentials are incorrect.

## AuthException

**Properties**

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| reason | String | Why authentication failed |

# IX. Testing

Primary testing will be done in one script that demonstrates multiple requirements. The output file will be included in the test folder.

Script Requirements:

1. Define permissions
2. Define roles
3. Add permissions
4. Create users
5. Add credentials
6. Add roles to users
7. Create resource roles
8. Add resource roles to users
9. Invoke a command requiring a permission from a role
10. Invoke a command requiring a resource role
11. Invoke a command from a microphone sensor event requiring a permission from a role

Script Specifics:

1. Executes all commands given in the sample script provided by the class. Then it gives a printout of all defined entitlements. It then prints all roles and users with their entitlements.
2. Adds the admin role to resident 1 then logs in as resident 1. Defines a city, a method that requires the admin role with the define city permission and shows the city. Defines a person, a method that requires the manage city permission in the admin role and shows the person.
3. Defines a microphone, which requires the devices permission. Creates a sensor event where resident 1 executes a command from the microphone. The command is defining resident 2. We show city 1 with resident 2 newly listed in the people section.

Other Scripts:

Script 2: the original sample script provided by the class but modified to fit this project’s syntax

Script 3: requires an entitlement that the user does not have, throws auth exception

Script 4: incorrect credential, throws auth exception

# X. Risks

Currently anyone with access to the controller can see the saved biometrics of the users, including the admins

# Appendix 1: List of Permissions

# auth\_user\_admin

# auth\_role\_entitlement

# auth\_resource\_admin

# scms\_control\_robot

# scms\_drive\_car

# scms\_manage\_city

# scms\_manage\_device

# Appendix 2: List of Roles

# adult\_role

# child\_role

# public\_admin

# admin\_role

# city1\_public\_admin

# city2\_public\_admin