Smart City Authentication Service Requirements

Author: Eric Gieseke

Date: 10/27/2020

Introduction

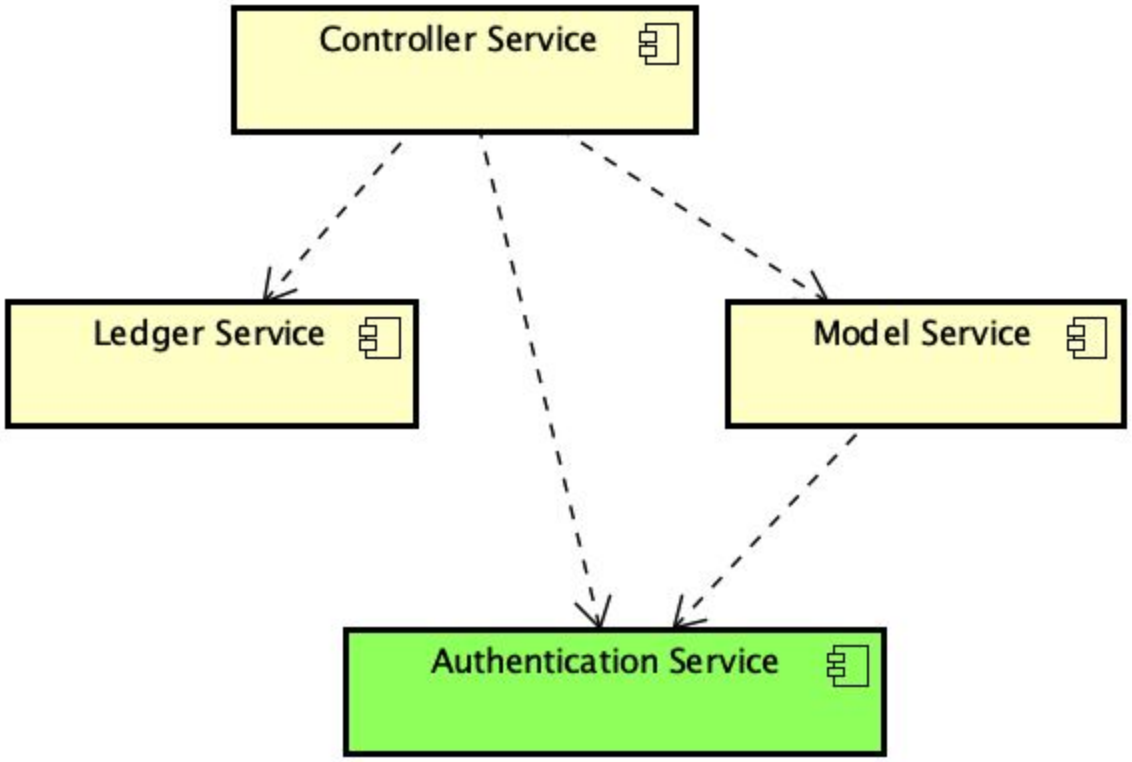
This document provides the requirements for the Smart City Authentication Service.

Overview

The Smart City System fully automates a modern city. The system allows the city inhabitants to control IoT devices through voice commands. Sensors monitor the location of the inhabitants. The Controller Service automatically updates the IoT devices based on the city's inhabitants and state. Inhabitants may issue voice commands to control devices.

Security is an essential aspect of the Smart City System. It is critical that the system only enables known inhabitants and trusted services to control IoT devices.

Smart City Integration



Caption: Smart City component diagram showing the Controller Service, Model Service, Ledger Service, and Authentication Service (highlighted in green) with their interdependencies.

The Authentication Service will be used to control access to the Model Service. The Model Service will be updated to use the Auth Token passed into each of the service methods and give this to the Authentication Service to validate that the associated user has the permission required to invoke the method.

When creating new inhabitants, create a corresponding user within the Authentication Service with a default voice print and face print.

When creating a new public administrator, create a Resource Role within the Authentication Service that binds the Person with the City and the appropriate Role.

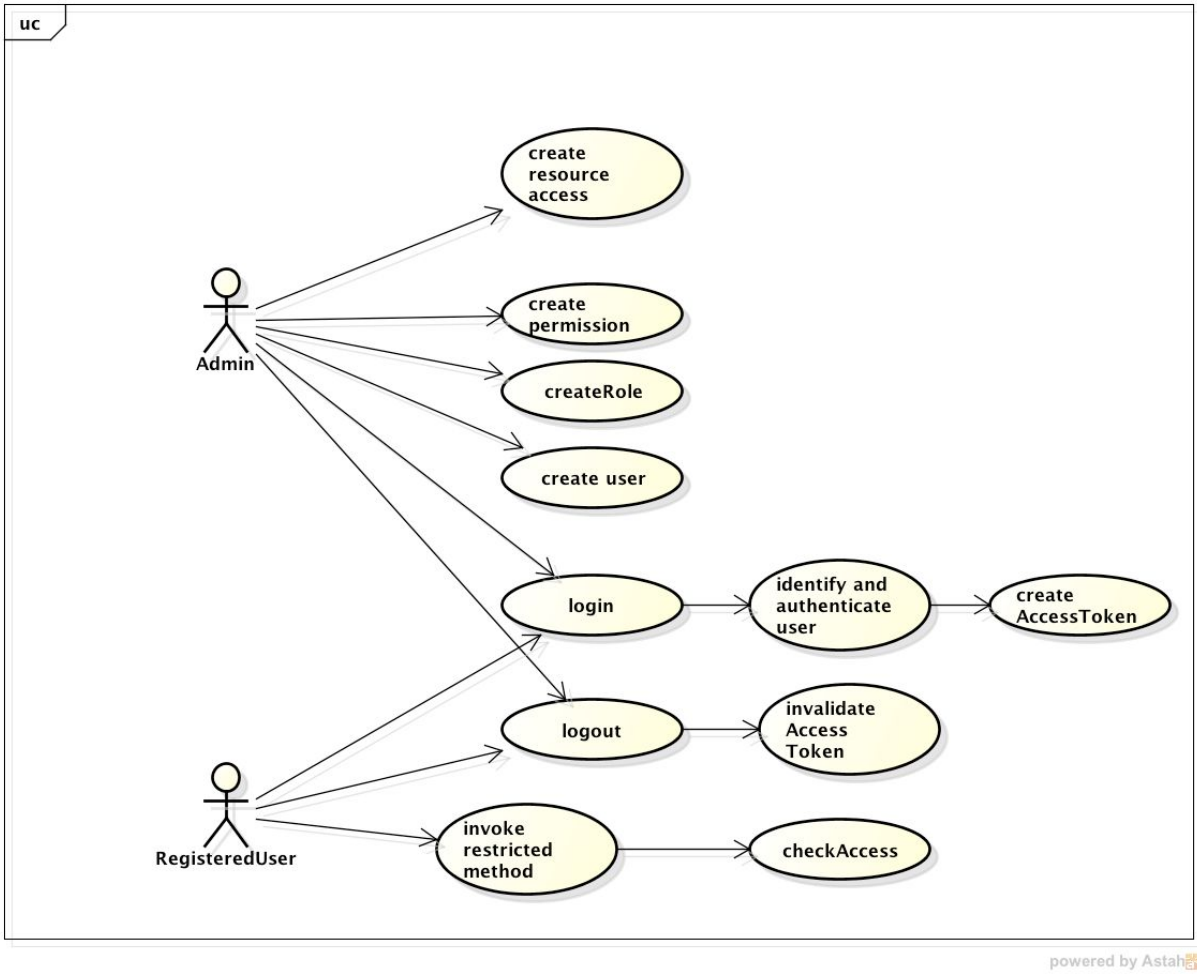
Extend the command processor to support the Authentication Service commands, as documented later in this document. Script commands must be preceded with a login request using an administrator username and password to obtain an Auth Token.

Update the Controller Service to request an authentication token from the Authentication Service. Then use the authentication token to access the Model Service API. For voice

commands received from a microphone sensor, the Controller Service will use either a voiceprint, faceprint, or another biometric identifier to obtain an auth token on behalf of the person making the request. The Controller Service will use the resulting Auth Token when calling the Model Service methods. In this way, the entitlements associated with the Person making the request are used to authorize or deny the request.

Authentication Service

The Authentication Service will assist in controlling access to the Model Service interface. The Authentication Service provides a central point for managing Users, Resources, Permissions, Roles, ResourceRoles, and Auth Tokens.



Caption: Use Case Diagram for Authentication Service

The Authentication Service supports two primary actors: Administrator and Registered User.

The Administrator is responsible for managing the Resources, Permissions, Roles, and Users maintained by the Authentication Service. The Administrator also provisions Cities and other entities within the Smart City system and has full access to all Smart City System resources. Voice or face prints identify city inhabitants.

Inhabitants use voice commands to interact with the Smart City System. The Authentication Service supports identifying and authenticating users using the user's voice or face print. When a Consumer issues a voice command, the Controller service sends the voiceprint and faceprint to the Authentication Service. The Authentication Service finds the user with a matching biometric identifier. It returns an Authentication Token, then passes the authentication token to the Model Service methods on behalf of the Inhabitants. In this way, only persons with the appropriate permissions can control IoT devices within the cities.

Authentication Service API

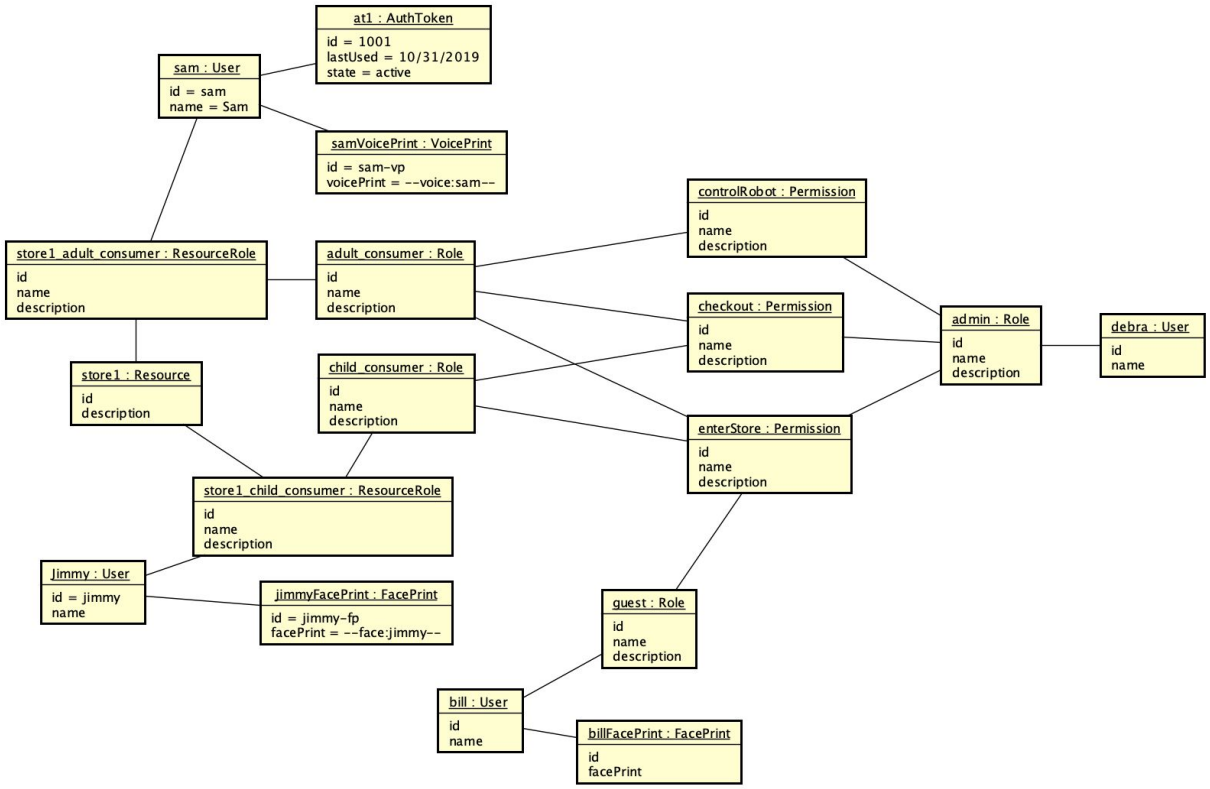
The Authentication Service API supports the following functions:

- Creating Resources:
  - A Resource represents a physical and logical entity, for example, an IoT Device.
  - A Resource has a unique identifier and a description.
- Creating Permissions:
  - Permissions represent an authorization required to access a resource or function of the Smart City system.
  - Permissions have a unique id, name, and description.
  - A User may be associated with zero or more permissions.
- 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.
- Creating Users:
  - Users represent persons of the Smart City system.
  - Users have an id, a name, and a set of Credentials. Credentials may include a username/password, voiceprint, faceprint, and other biometric identities. Hash the credentials to secure their use.
  - Users are associated with 0 or more entitlements (i.e., Roles or Permissions).
- Authentication
  - The Authentication process provides users AuthTokens that can then be used to access restricted Service Methods.
  - If authentication fails, an Authentication Exception should be thrown.
  - If authentication succeeds, an AuthToken is created and returned to the caller.
  - The AuthToken binds the User to a set of permissions that can be used to grant or deny access to restricted methods.
  - AuthTokens can timeout with inactivity.
  - AuthTokens have a unique id, an expiration time, and a state (active or expired).

    - Auth tokens are associated with a User and a set of Permissions.
        - Login:
            - Login accepts a User's credentials (username, password).
            - Validate that the username exists, and then that the hash of the password matches the known hashed password.
        - VoicePrint
            - Voiceprint supports authentication through voice recognition
            - Simulate a voiceprint using a string in this format: "`voice-print =voiceprint-`<username>". For example, the voiceprint for Jane is "`voice-print='voiceprint-jane'`".
            - The voice print signature is sufficient for identifying and authenticating a user.
        - FacePrint
            - Faceprint supports authentication through face recognition
            - Simulate a faceprint using a string in this format: "`face-print='faceprint-`<username>". For example, the faceprint for Jane is "`face-print='faceprint-jane'`".
- Logout:
    - Logout marks the given Auth Token as invalid.
    - Subsequent attempts to use the AuthToken should result in an InvalidAuthTokenException.
- Smart City Model Service:
    - All methods defined within the Model Service should accept an AuthToken
    - Each method should validate that the AuthToken is non-null and non-empty.
    - The method should pass the AuthToken to the Authentication Service with the required permission.
    - The Authentication Service should check to make sure that the AuthToken is active and within the expiration period. Then, check that the user associated with the AuthToken has the permission required by the method.
    - The Authentication Service should throw an 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, Resource Role, AccessToken, and User Credentials are related.



Caption: Sample instance diagram for the Authentication Service.

Restricted methods include restricted methods on the Authentication and Model Services.