MOBILE APPLICATION STORE AUTHENTICATION SERVICE

DESIGN DOCUMENT

NOVEMBER 22, 2013

FANXING MENG
Reviewer: Jayaprakash Ganta

Table of Contents

Introduction

This document defines the design for the Mobile Application Store Authentication Service. Mobile application stores provide contents including applications, ringtones and wallpaper for mobile device users. Authentication service enables administrators to set permission to different categories of users to ensure store content safety against unauthorized stack.

Overview

In this phase we have developed the authentication service of the app store. The authentication service allows administrators to define services, permissions for each restricted operation, roles that users could assign to, users and their credentials, and manage entitlement among roles and users. The authentication service starts with only an admin user having root privilege. The administrator must first log on to this user to set up necessary permissions and authentication_admin users. After which he/she should change the admin password and log on to a regular (specific) admin account using user name and password to manage the services available. For each login, the user receives a token; he/she should pass the token as an argument to access restricted functions. If the user does not have the necessary permission, he/she would get an exception specific to the request.

Requirements

The functional requirements are that only an authentication administrator is able to create services with that aggregates a set of permissions; the admin can create permissions for each restricted function within a service; he/she creates roles that are composites of permissions and roles, and assign them to users, which the admin also has to create. Any registered users may log in the system using user name and password and receive a token for future authentication in the current session if log in succeeds; the token has a certain expiration time to remain active. Upon logout, the assigned token becomes invalid. To ensure restricted access among all services, each of these restricted functions should validate a passes token in authentication service to check for required permission.

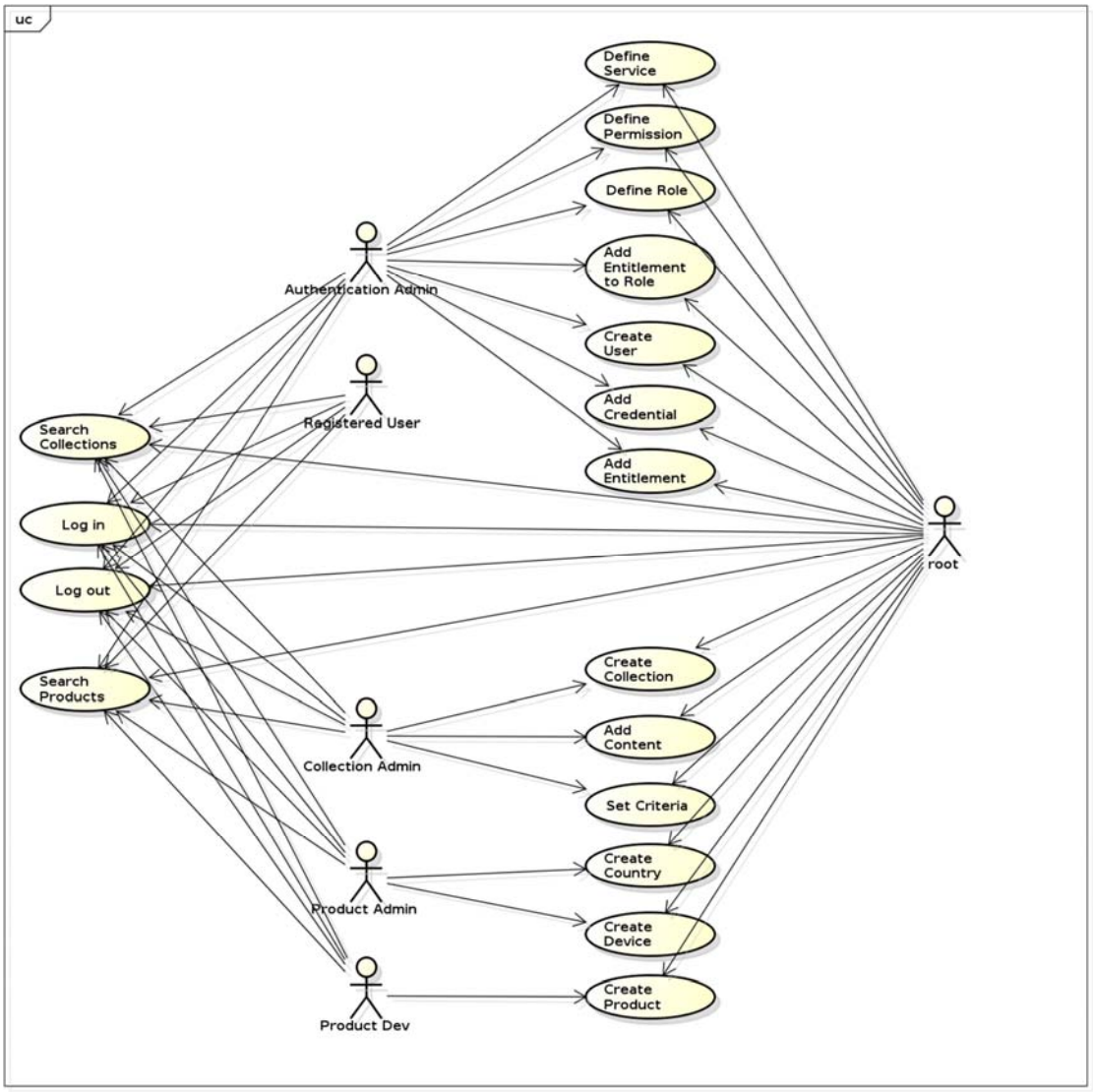
The structural requirements are: Service, Permission, and Role all have fields ID, name, and description; Service contains Product API, Collection Service, and Authentication Service; A Service may have one or more permissions while a User may have zero or more permissions or roles; A User has an ID, name, and a set of credentials, each contains a user name and a hashed password; A Token has an ID, expiration time, and a state; it binds a User to a set of Permissions.

Use Cases

Current design supports the following use cases:

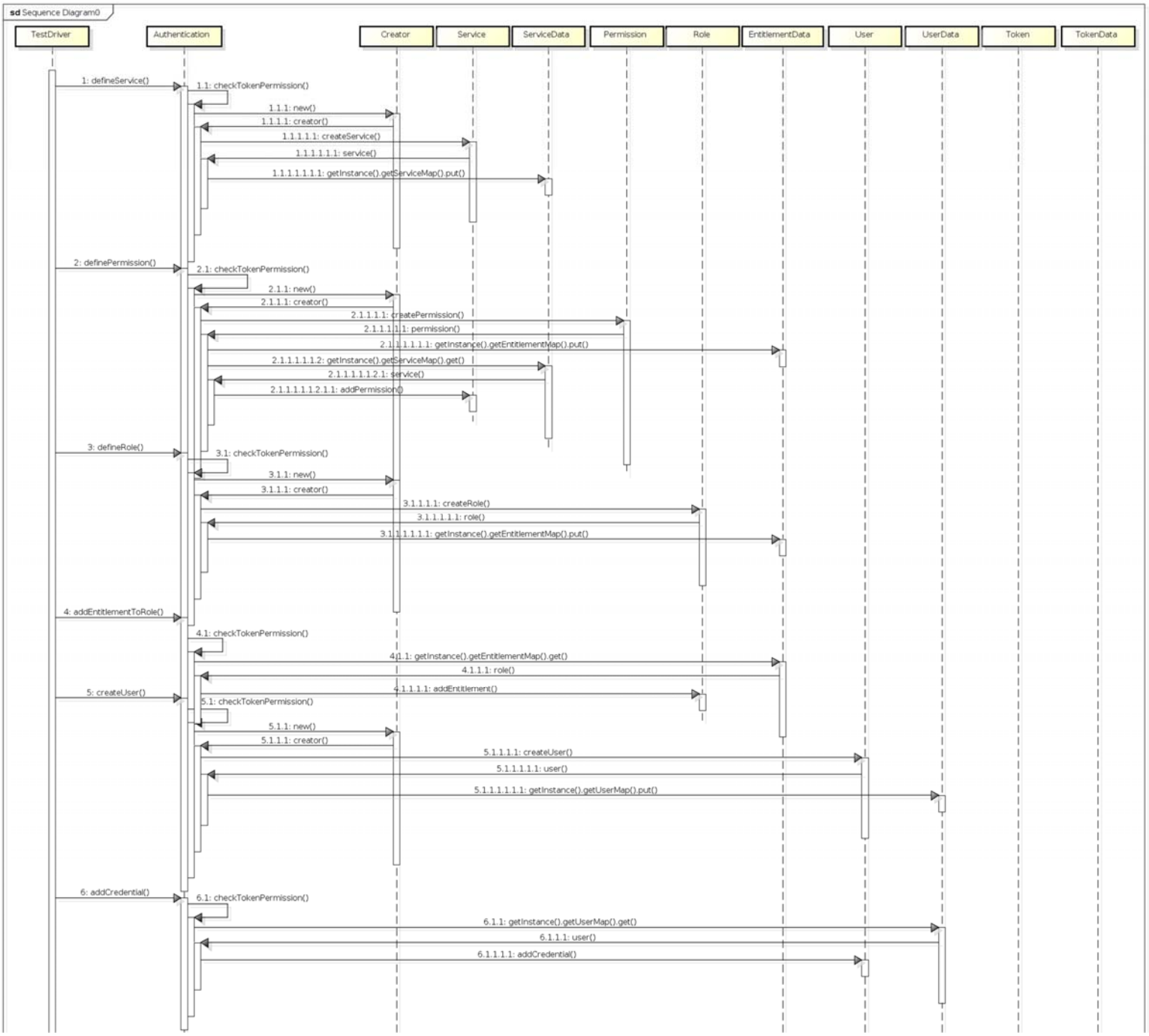
The root user has all available permissions. All users and administrators can log in, log out, and search collections and products.

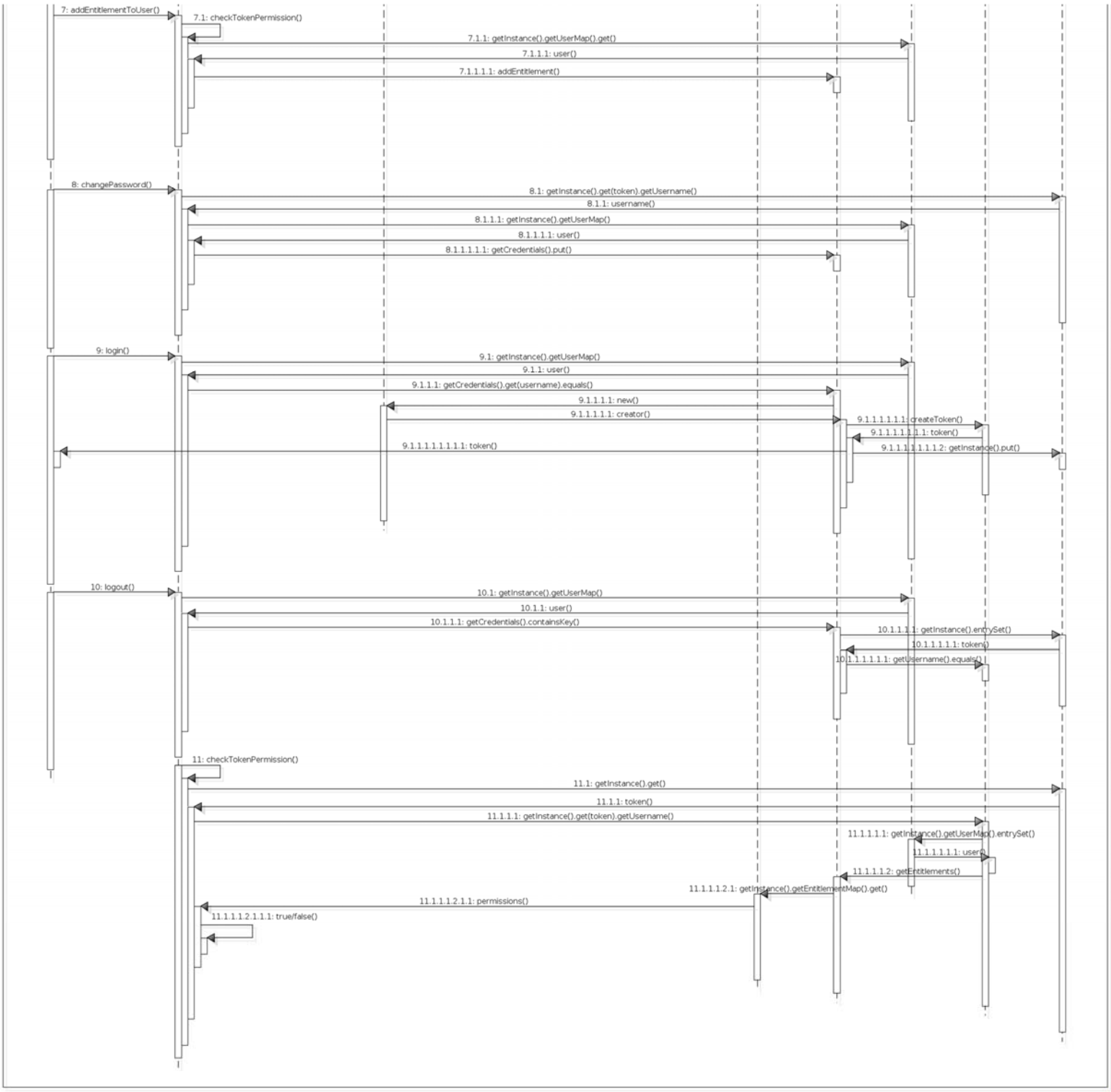
Among all restricted functions, the authentication admin has permission to the seven functions affiliated with authentication service (also the authentication admin role) at the top of the diagram; collection admin can create collection, add content to static collection and set search criteria for dynamic collection; product admin can create country and devices; finally product developer can create product.



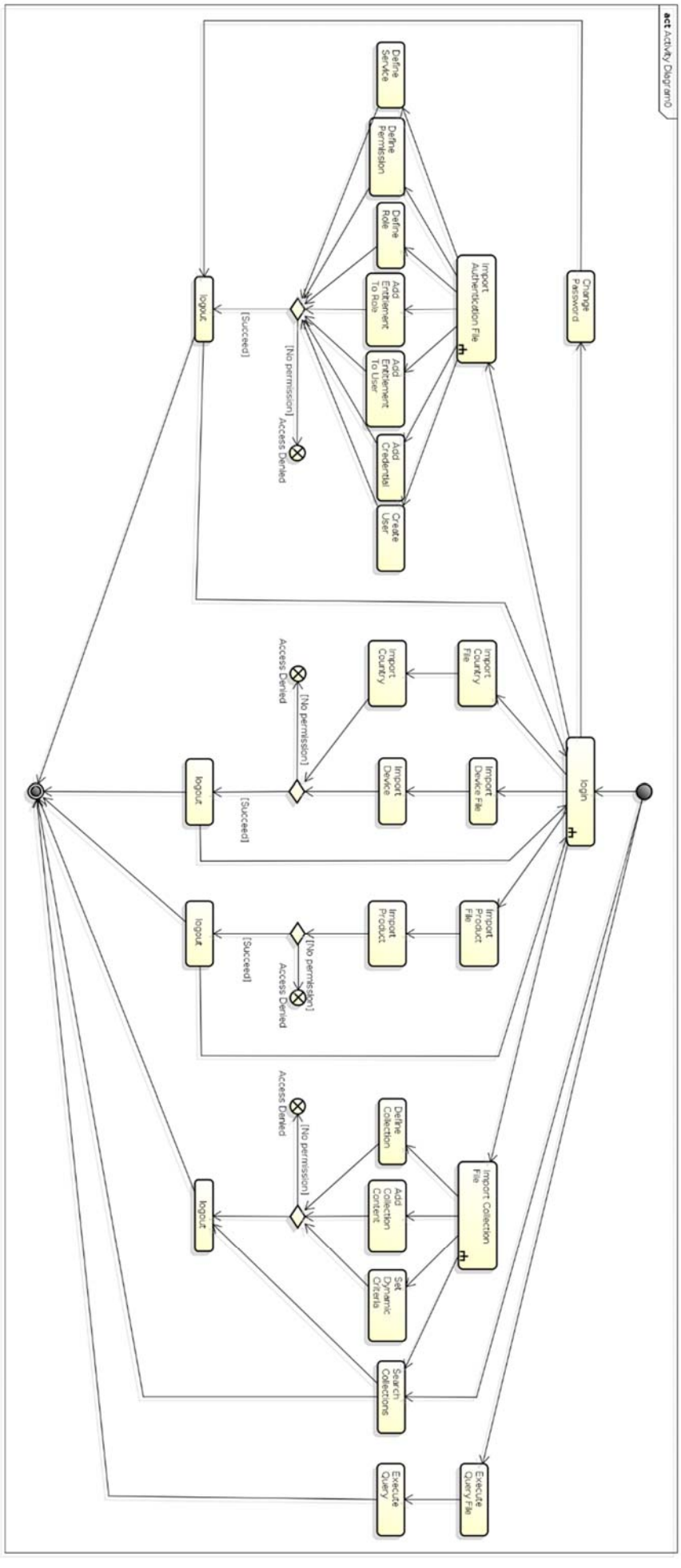
Sequence Diagram

Each class will cooperate with other classes as shown below:



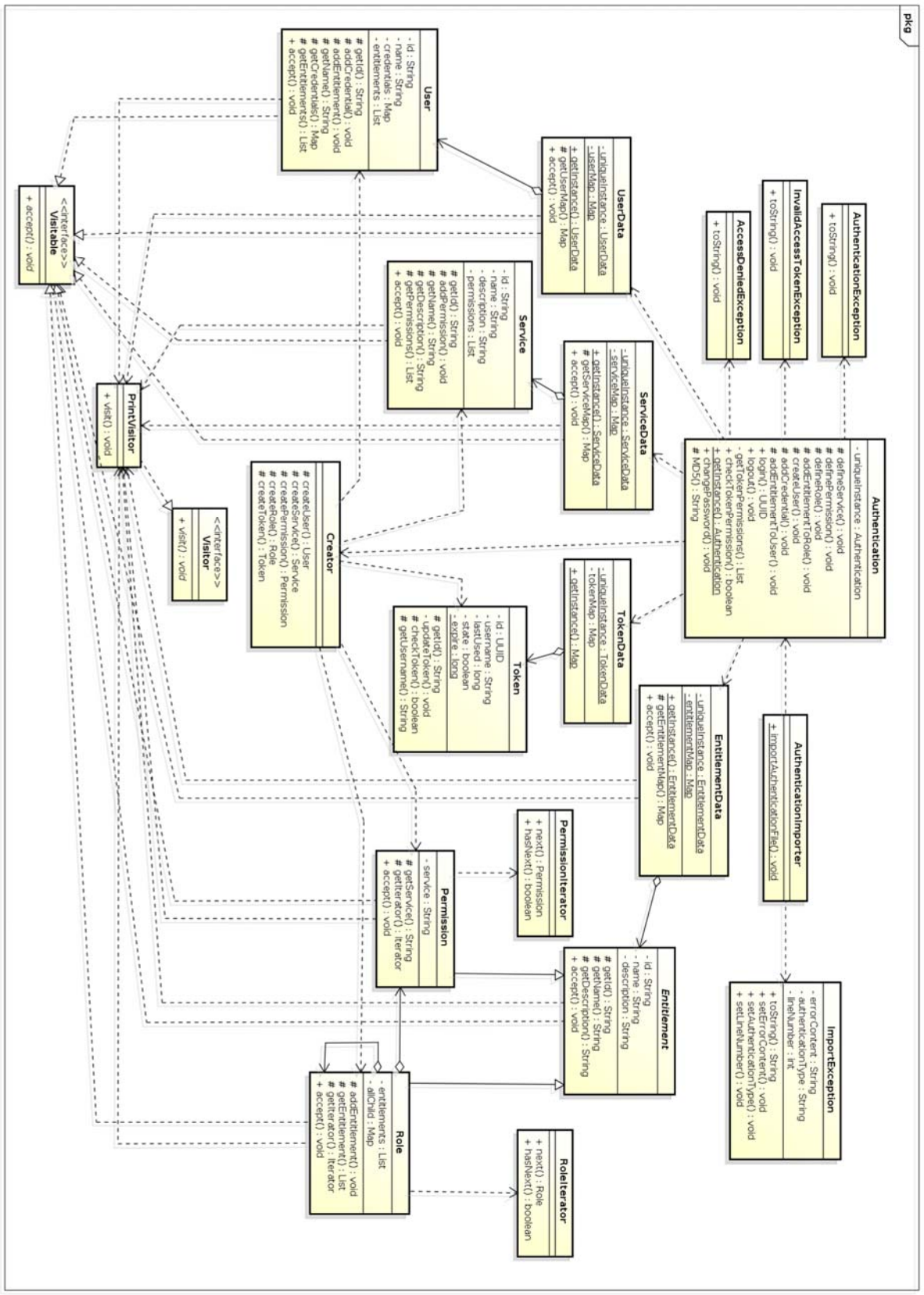


Activity Diagram



Implementation

Class Diagram



Class Dictionary

This section specifies the class dictionary for the authentication service. The classes are defined within the package “cscie97.asn4.ecommerce.authentication”.

AccessDeniedException

The AccessDeniedException class provides exception to handle situations where a username/password is invalid or missing

Methods

Method Name	Signature	Description
toString	(void):void	Public method for formatting the output information of this type of exception as "Access Denied: message"

Authentication

The Authentication class provides an instance of the authentication service with functions necessary to create, manage, and test all authentication functions.

Methods

Method Name	Signature	Description
getInstance	(void):Authentication	Public method for getting the unique instance of the authentication service.
defineService	(newId:String, newName:String, newDescription:String, token:UUID, permission:String):void	Protected method for defining a service. Checks for token permission and duplicate service ID. Throws AuthenticationException on duplicate ID and InvalidAccessTokenException on token not having correct permission.
definePermission	(serviceId:String, permissionId:String, newName:String, newDescription:String, token:UUID, permission:String):void	Protected method for defining a permission. Checks for token permission, duplicate permission ID, and invalid service ID. Throws AuthenticationException on duplicate or invalid ID and InvalidAccessTokenException on token not having correct permission.

defineRole	(newId:String, newName:String, newDescription:String, token:UUID, permission:String):void	Protected method for defining a role. Checks for token permission and duplicate role ID. Throws AuthenticationException on duplicate ID and InvalidAccessTokenException on token not having correct permission.
addEntitlementToRole	(roleId:String, entitlementId:String, token:UUID, permission:String):void	Protected method for adding an entitlement to role. Checks for token permission, duplicate/invalid entitlement ID, and invalid role ID. Throws AuthenticationException on duplicate or invalid ID and InvalidAccessTokenException on token not having correct permission.
createUser	(newId:String, newName:String, token:UUID, permission:String):void	Protected method for creating a user. Checks for token permission and duplicate user ID. Throws AuthenticationException on duplicate ID and InvalidAccessTokenException on token not having correct permission.
addCredential	(userId:String, newName:String, newPassword:String, token:UUID, permission:String):void	Protected method for adding credential to a user. Checks for token permission, duplicate user name, and invalid user ID. Throws AuthenticationException on duplicate or invalid ID and InvalidAccessTokenException on token not having correct permission.
changePassword	(token:UUID, password:String):void	Public method for changing a user's password. Checks for token existence and validity. Throws AccessDeniedException on non-existing token and InvalidAccessTokenException on expired token.

addEntitlementToUser	(userId:String, entitlementId:String, token:UUID, permission:String):void	Protected method for adding an entitlement to a user. Checks for token permission, duplicate/invalid entitlement ID, and invalid user ID. Throws AuthenticationException on duplicate or invalid ID and InvalidAccessTokenException on token not having correct permission.
login	(username:String, password:String):UUID	Public method for logging in. Checks for user name existence and password validity. Throws AccessDeniedException for either false.
logout	(username:String):void	Public method for logging out. Checks for user name existence. Throws AccessDeniedException for non-existing user name.
getTokenPermissions	(token:UUID):List<String>	Private method for getting token's permissions. Returns empty list if there's no permission associated with it.
checkTokenPermission	(token:UUID, permission:String):Boolean	Public method for checking token against a permission. Checks for token existence and validity. Throws AccessDeniedExceptionon non-existing token and InvalidAccessTokenException on expired token.
MD5	(md5:String):String	Protected method for computing MD5 value from a string. Throws NoSuchAlgorithmException on error.

Properties

Property Name	Type	Description
uniqueInstance	Authentication	Private field maintaining the unique instance of the authentication service.

AuthenticationException

The AuthenticationExceptionclass provides exception to handle situations where a field is invalid or missing.

Methods

Method Name	Signature	Description
toString	(void):void	Public method for formatting the output information of this type of exception as " Authentication Exception: message"

AuthenticationImporter

The importer is responsible for reading authentication information from .csv file. The importer calls authentication functions to dispatch operation on each line of code.

Methods

Method Name	Signature	Description
importAuthenticationFile	(filename:String, token:UUID):void	Public method for reading authentication information from .csv file line by line and calls functions in Authentication class to import data. Checks for operation type and argument number. Throws ImportException on error parsing the file.

Creator

The creator is responsible for creating service, user, permission, role, and token objects.

Methods

Method Name	Signature	Description
createService	(newId:String, newName:String, newDescription:String):Service	Protected method for creating a service.
createUser	(newId:String, newName:String):User	Protected method for creating a user.
createPermission	(newId:String, newName:String, newDescription:String, newServicen:String): Permission	Protected method for creating a permission.
createRole	(newId:String, newName:String, newDescription:String):Role	Protected method for creating a role.

createToken	(newId:UUID, newName:String, time:long):Token	Protected method for creating a token.

Entitlement

The Entitlement abstract class implements the Visitable interface and provides the fields and methods that are common to both roles and permissions.

Methods

Method Name	Signature	Description
getId	(void):String	Protected method for returning the entitlement ID.
getName	(void):String	Protected method for returning the entitlement name.
getDescription	(void):String	Protected method for returning the entitlement description.
getIterator	Iterator	Protected method for getting a recursive iterator of entitlements.
accept	(visitor:Visitor):void	Public method for accepting a visitor to the entitlement.

Properties

Property Name	Type	Description
id	String	Protected id field for the entitlement.
name	String	Protected name field for the entitlement.
description	String	Protected description field for the entitlement.

EntitlementData

The EntitlementData class implements the Visitable interface and provides an aggregation of Entitlement objects enabling visitors to traverse, initialized with root permission and role.

Methods

Method Name	Signature	Description
getInstance	(void):EntitlementData	Public method for getting the unique instance of the EntitlementData class.

getEntitlementMap	(void): Map<String, Entitlement>	Protected method for getting the entitlement map of the EntitlementData class.
accept	(visitor:Visitor):void	Public method for letting the visitor traverse through the entire entitlement map.

Associations

Association Name	Type	Description
entitlementMap	Map<String, Entitlement>	Private association for listing all created entitlements.

Properties

Property Name	Type	Description
uniqueInstance	EntitlementData	Private unique instance of the EntitlementData class.

ImportException

The ImportExceptionclass provides exception to handle situations where there is an error in the input authentication file.

Methods

Method Name	Signature	Description
toString	(void):String	Public method that formats the output information of this type of exception as “error importing authentication type: authenticationType at line number lineNumber with error errorContent”
setErrorContent	(errorContent:String):void	Public method that sets the error content using passed-in argument.
setAuthenticationType	(authenticationType:String):void	Public method that sets the authentication type using passed-in argument.
setLineNumber	(lineNumber:int):void	Public method that sets the line number using passed-in argument.

Properties

Property Name	Type	Description

errorContent	String	Private content field for the error.
authenticationType	String	Private type field for the authentication exception.
lineNumber	int	Private line number field storing where the error occurred.

InvalidAccessTokenException

The InvalidAccessTokenExceptionclass provides exception to handle situations where the token is invalid for a certain permission.

Methods

Method Name	Signature	Description
toString	(void):void	Formats the output information of this type of exception as " Invalid access token for: message"

Properties

Property Name	Type	Description
errorContent	String	Private content field for the error.

Permission

The Permission class implements the Visitable interface and provides the fields and methods for permissions.

Methods

Method Name	Signature	Description
getService	(void):String	Protected method for getting the service ID associated with the permission.
getIterator	Iterator	Protected method for getting a null iterator of permission.

Properties

Property Name	Type	Description
service	String	Private service id field for the permission.

PermissionIterator

The Class PermissionIterator that implements java's iterator but returns null and false for permissions as they are the end of the entitlement tree.

Methods

Method Name	Signature	Description
hasNext	(void):boolean	Public method for checking whether there is a next element at the end of entitlement tree, which is false.
next	(void):Object	Public method for getting the next element at the end of entitlement tree, which has none.
remove()	(void):void	Not supported.

PrintVisitor

The PrintVisitor class implements Visitor interface and prints out one message for each object it visits.

Methods

Method Name	Signature	Description
visit	(user:User):void	Public method for visiting a user.
visit	(permission:Permission):void	Public method for visiting a permission.
visit	(role:Role):void	Public method for visiting a role.
visit	(service:Service):void	Public method for visiting a service.
visit	(entitlement:Entitlement):void	Public method for visiting an entitlement.
visit	(serviceData:ServiceData):void	Public method for visiting all ServiceData.
visit	(userData:UserData):void	Public method for visiting all UserData.
visit	(entitlementData:EntitlementData):void	Public method for visiting all EntitlementData.

Role

The Role class implements the Visitable interface and provides the fields and methods for roles.

Methods

Method Name	Signature	Description
addEntitlement	(String):void	Protected method for adding an affiliated entitlement to the role. Checks for duplicate entitlement id. Throws AuthenticationException for duplicate entitlement id.
getEntitlement	(void):List<String>	Protected method for getting all entitlements affiliated with the role.
getIterator	Iterator	Protected method for getting a compound iterator of all child entitlements.

Associations

Association Name	Type	Description
entitlements	List<String>	Private association for listing the entitlements affiliated with the role.
allChild	Map<String, Entitlement>	Private association for temporarily storing the entitlement objects in the list for iterating.

RoleIterator

The Class RoleIterator that implements java's iterator but use an auxiliary stack to traverse the entitlement tree.

Methods

Method Name	Signature	Description
hasNext	(void):boolean	Public method for checking whether there is a next element at the end of entitlement tree.
next	(void):Object	Public method for getting the next element at the end of entitlement tree.
remove()	(void):void	Not supported.

Service

The Service class implements the Visitable interface and provides the fields and methods that are common to Service objects.

Methods

Method Name	Signature	Description
getId	(void):String	Protected method for returning the service ID.
getName	(void):String	Protected method for returning the service name.
getDescription	(void):String	Protected method for returning the service description.
addPermission	(newPermission:String):void	Protected method for adding a permission to the service. Checks for duplicate permission id. Throws AuthenticationException for duplicate permission id.
getPermissions	(void):List<String>	Protected method for getting all permissions associated with the service.
accept	(visitor:Visitor):void	Public method for accepting a visitor to the service.

Associations

Association Name	Type	Description
permissions	List<String>	Private association for listing the permissions associated with the service.

Properties

Property Name	Type	Description
id	String	Private id field for the service.
name	String	Private name field for the service.
description	String	Private description field for the service.

ServiceData

The ServiceData class implements the Visitable interface and provides an aggregation of Service objects enabling visitors to traverse.

Methods

Method Name	Signature	Description
getInstance	(void):ServiceData	Public method for getting the unique instance of the ServiceData class.
getServiceMap	(void): Map<String, Service>	Protected method for getting the service map of the ServiceData class.
accept	(visitor:Visitor):void	Public method for letting the visitor traverse through the entire service map.

Associations

Association Name	Type	Description
serviceMap	Map<String, Service>	Private association for listing all created services.

Properties

Property Name	Type	Description
uniqueInstance	ServiceData	Private unique instance of the ServiceData class.

Token

The Token class provides the fields and methods that are common to Token objects.

Methods

Method Name	Signature	Description
getId	(void):UUID	Protected method for returning the token ID.
getName	(void):String	Protected method for returning the user name of the token.
updateToken	(time:long):void	Private method for updating the token's state and last used time.
checkToken	(void):boolean	Protected method for checking the token's current status, true for active and false for expired.

Properties

Property Name	Type	Description

id	UUID	Private id field for the token.
username	String	Private user name field for the token.
lastUsed	long	Private field for recording the last used time.
state	boolean	Private state field for the token, true for active and false for expired.
expire	long	Private static final field for the expiration time length.

TokenData

The TokenData class provides an aggregation of Token objects.

Methods

Method Name	Signature	Description
getInstance	(void):Map<UUID, Token>	Public method for getting the token map of the unique instance of the TokenData class.

Associations

Association Name	Type	Description
tokenMap	Map<UUID, Token>	Private association for listing all created tokens.

Properties

Property Name	Type	Description
uniqueInstance	TokenData	Private unique instance of the TokenData class.

User

The User class implements the Visitable interface and provides the fields and methods that are common to User objects.

Methods

Method Name	Signature	Description
getId	(void):String	Protected method for returning the user ID.
getName	(void):String	Protected method for returning the user name.

addCredential	(login:String, passwordMD5:String):void	Protected method for adding a set of credential to the user. Checks for duplicate user name. Throws AuthenticationException for duplicate user name.
getCredentials	(void):Map<String, String>	Protected method for getting all credentials associated with the user.
addEntitlement	(login:String, passwordMD5:String):void	Protected method for adding an entitlement to the user. Checks for duplicate entitlement id. Throws AuthenticationException for duplicate entitlement id.
getEntitlements	(void):Map<String, String>	Protected method for getting all entitlements associated with the user.
accept	(visitor:Visitor):void	Public method for accepting a visitor to the user.

Associations

Association Name	Type	Description
credentials	Map<String, String>	Private association for listing the credentials (user names and hashed passwords) of the user.
entitlements	List<String>	Private association for listing the entitlements assigned to the user.

Properties

Property Name	Type	Description
id	String	Private id field for the user.
name	String	Private name field for the user.

UserData

The UserData class implements the Visitable interface and provides an aggregation of User objects enabling visitors to traverse, initialized with an admin user with root role.

Methods

Method Name	Signature	Description

getInstance	(void):UserData	Public method for getting the unique instance of the UserData class.
getUserMap	(void): Map<String, User>	Protected method for getting the user map of the UserData class.
accept	(visitor:Visitor):void	Public method for letting the visitor traverse through the entire user map.

Associations

Association Name	Type	Description
userMap	Map<String, User>	Private association for listing all created users.

Properties

Property Name	Type	Description
uniqueInstance	UserData	Private unique instance of the UserData class.

Visitable

The Visitable interface provides common method for accepting a visitor.

Methods

Method Name	Signature	Description
accept	(visitor:Visitor):void	Public method for letting the visitor visit a class object.

Visitor

The Visitor interface provides common method for visiting various class objects.

Methods

Method Name	Signature	Description
visit	(user:User):void	Public method for visiting a user.
visit	(permission:Permission):void	Public method for visiting a permission.
visit	(role:Role):void	Public method for visiting a role.
visit	(service:Service):void	Public method for visiting a service.

visit	(entitlement:Entitlement):void	Public method for visiting an entitlement.
visit	(serviceData:ServiceData):void	Public method for visiting all ServiceData.
visit	(userData:UserData):void	Public method for visiting all UserData.
visit	(entitlementData:EntitlementData):void	Public method for visiting all EntitlementData.

Implementation Details

There are several details that are worth mentioning.

Singleton pattern is enforced in Authentication, UserData, ServiceData, EntitlementData, and TokenData classes, using private constructor and static initialization of the instance. Although this is not prettier than just using static maps for User, Service, Entitlement and Token data with static functions in Authentication class, it becomes helpful when initialization needs to add the admin user and root permission.

Factory pattern is applied to creating User, Service, Role, Permission, and Token objects. But this only added a layer of redundancy and did not provide helpful abstraction.

Visitor pattern is purely used as a traversing-printing functionality because not too much can be done from just an object argument.

Since there is vaguely any constraint on how to assign exception types, the choice in this implementation is somewhat arbitrary: when any object that should exist but did not, or if an operation induces duplicate items, we will throw an Authentication Exception; whenever checking token for a specific permission returns false, we throw an Invalid Access Token Exception; if a user name does not exist or the password is incorrect when logging in/out or a token does not exist, we throw an Access Denied Exception.

To simplify the overall logic in iterating entitlements, we use two types of iterators, one for roles which may contain other entitlements, and a null iterator for permissions which cannot contain any other entitlements. In this way we achieve simple traversal by just using a stack and eliminates the need to detect object types at every point. To help traversing, we put a temporary map of entitlements inside the Role class and populate it from the list of entitlement ID's before getting a compound iterator out of them.

Testing

Implement a test driver class called TestDriver that has a static main() method. The main() method should accept 5 parameters: an input authentication file, an input countries data file, a devices data file, a products data file, and a collection file. After the admin logs in, the main method will call the AuthenticationImporter.importAuthenticationFile(), passing in the name of the authentication file and the admin's token. The admin then logs out and let a newly created product admin to log in and import country file and device file. Then a product developer user logs in and adds product data. Finally a collection admin logs in and imports collection data.

When all files have been successfully imported, the UserData, ServiceData, and EntitlementData calls accept() over all their members to allow for visitors to traverse through the collections and print as they go.

The TestDriver class should be defined within the package “cscie97.asn4.test”.