I-o-T Solutions Project Report

Group 9
Omkar Desai (ord150030)
Shubhum Batra (sxb157630)

The University of Texas at Dallas

Table of Contents

1. Executive Summary

In this ever growing technological troposphere evolution is the need of the hour. Digital, centralized home/office solutions are the epitome of evolution and with an already existent consumer base we should capitalize on the opportunity of delivering a new I-o-T compliant systems.

The system would enable digital access to all home utilities using central terminals accessible at multiple nodes. User data like users logging in and out of system, temperature settings, lightning, etc. would be gathered and utilized for security and optimization purposes.

The proposed system would be controlled using a mobile application and other monitors located at the site. Warnings regarding open doors, kitchen appliances left turned on or any other forms of home safety hazards would be provided to all the registered users via a notification on the mobile app.

Utilizing the user data, we should be able to track and control the amount of energy consumed thereby reducing the cost of utilities. To implement the proposed system, we need to introduce a new line of internet ready products which would use new technologies.

2. System Proposal and Problem Statement

Background

Company X currently offers traditional home and office solutions. In this ever growing technological sphere every company needs to evolve. Digital, centralized home/office solutions are the epitome of evolution and with an already existent consumer base we should capitalize on the opportunity of delivering a new IOT compliant systems.

Goals

• To develop a centralized system for controlling all appliances.

• Develop an accounting system which would track the usages.

• Provides remote home safety and security.

Scope

This system should enable digital access to all home utilities using central terminals accessible at multiple nodes. Usage data like users logging in and out of system, temperature settings, lightning, etc. would be gathered and utilized for security and optimization purposes.

The proposed system would be controlled using a mobile application and other monitors located at the site. Warnings regarding open doors, kitchen appliances left turned on or any other forms of home safety hazards would be provided to all the registered users via a notification on the mobile app.

Key Stakeholders

Client: John Doe

Sponsor: John Doe

Project manager: Omkar Desai

Project team members: Shubham Batra, Omkar Desai

Project Milestones

• Requirements Documentation.

• Central System Development.

• UAT of the System.

Project Budget

Total Budget: $400 million

Constraints, Assumptions, Risks and Dependencies

Constraints: Hardware Development for the new Internet ready appliances.

Assumptions: Assuming all the houses have internet connection. Non Digital appliances would be made internet ready using the new microprocessor. Customers are going to adapt to the central terminals and mobile app provided for controlling the appliances.

Risks and Dependencies: Adopting a relatively new technology. Meeting Safety Regulations for different environments.

Deliverables

• Internet ready appliances.

• Central control terminals.

• Remote control mobile application.

• End User Documentation(User-Manual)

Project Success Criteria

• Centralized System is able to control all the appliances.

• The system is able to monitor and track usage of all the utilities.

• Implemented features are Safety Compliant

3. Requirements Definition

a. Functional Requirements

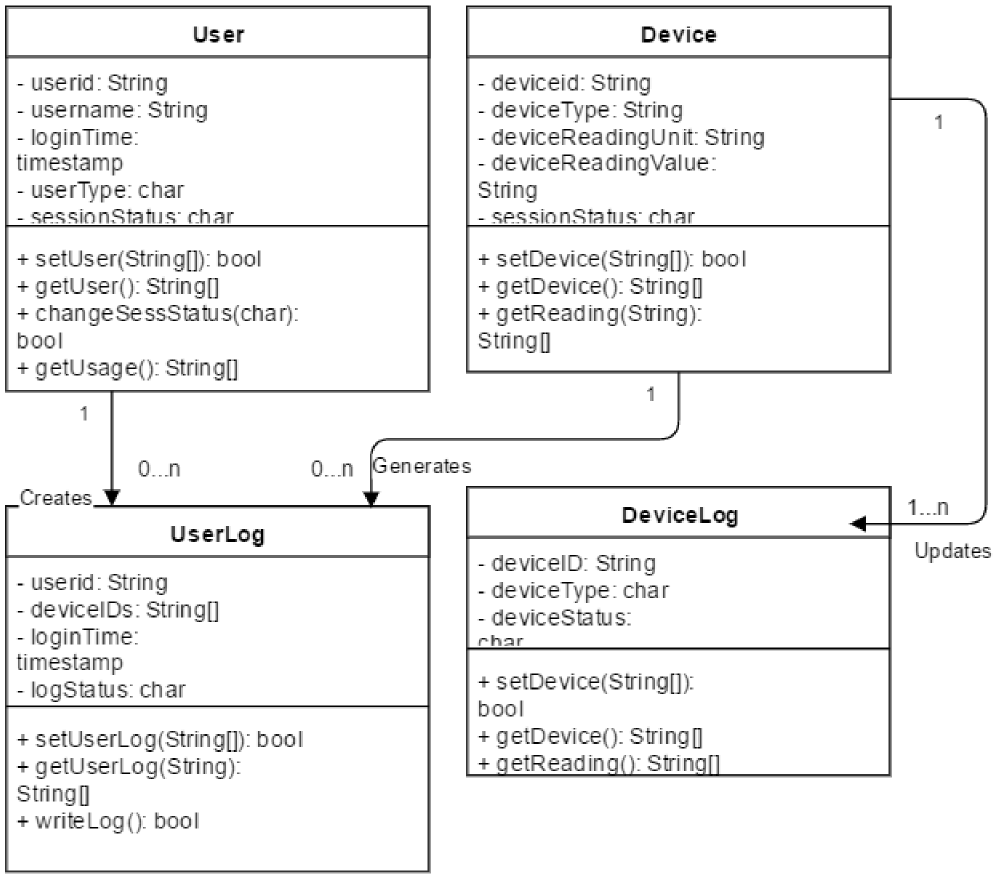
i. Each terminal can be used only after user authentication. Terminals are usable by users and guests. Users have full utility and security control except for root/administrator access. Guest users will not be able to access security features.
ii. All sessions should timeout in 10 minutes.
iii. System administrator should be able to add new users using any of the terminals after a valid administrator login. New users must confirm entry into the system before administrator session terminates.
iv. System administrator should be able to view current status of all connected devices along with usage reports for user specified periods.

b. Non-functional Requirements

i. Each command send to a device should be executed within 1 second. This is a **performance** requirement.
ii. Device functionalities should be accessible through any terminal (local or mobile) and should be contained within a menu of maximum depth 3, that is, any device command should be accessible within 3 clicks of the terminal. This is a **usability** requirement.
iii. Terminals should be heat and water resistant so as to not fail in the outdoor, kitchen, bathroom or other extreme environments. This is a **reliability** requirement.
iv. Critical system functioning should not fail in the case of power loss or network issues. Local terminals, devices and security must be connected via a backup hardline with their own backup power. This is a **reliability** requirement.

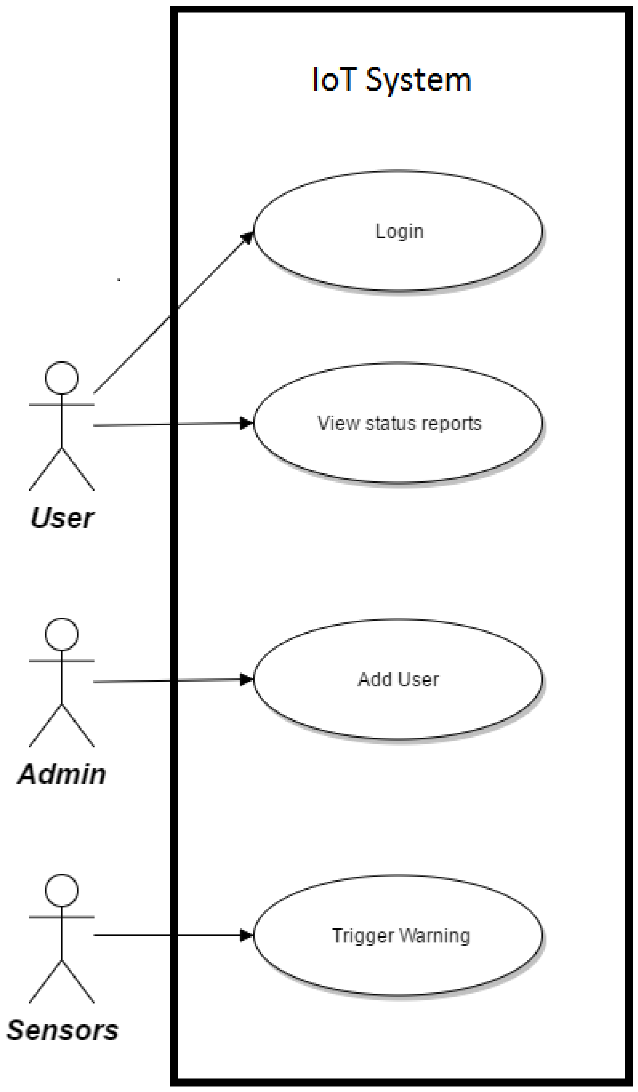
4. Structural Models

Class Diagram



5. Behavioral Models

a. Use Case Diagram



b. Use Case Description

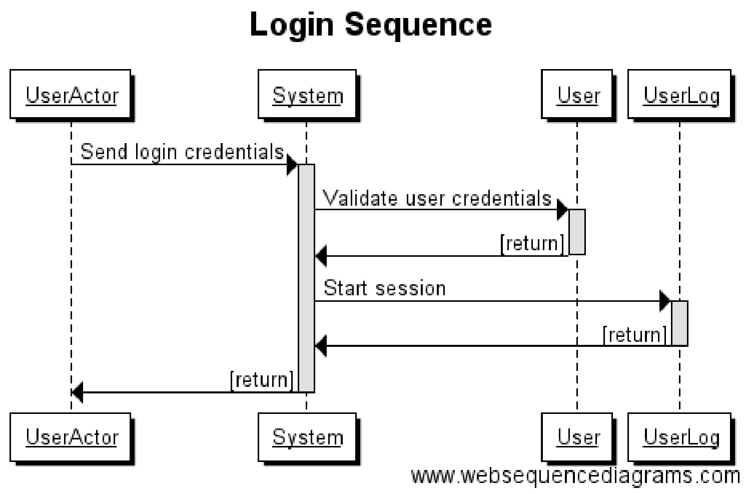
Use Case Name:	Login
Primary Actor:	User
Brief Description:	Allows user to login to system
Stakeholders:	User
Trigger:	When user accesses the terminal or the remote application.
Normal flow of events:	1. User sends login credentials. 2. System responds to user.
Sub-flows:	1. System verifies credentials. 2. System begins a session of ten minutes.
Alternate/Exception flow:	1. User is already logged in, system returns warning and refreshes session. 2. User cannot be verified, system returns error.

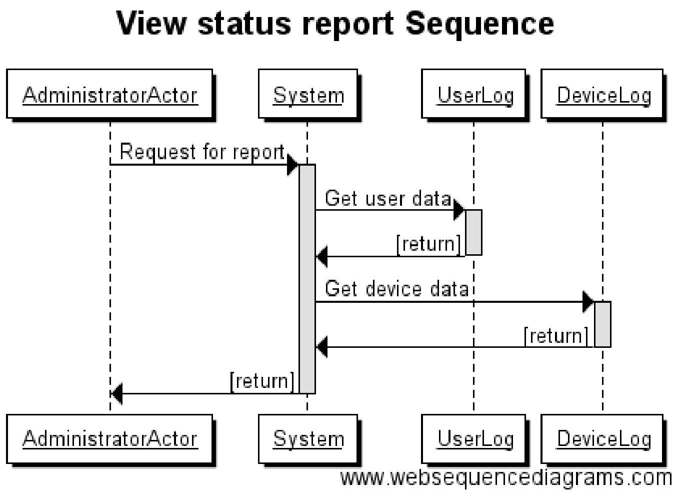
Use Case Name:	View Status Reports.
Primary Actor:	User
Brief Description:	Allows the primary user or administrator access to reports.
Stakeholders:	User.
Trigger:	When administrator requests for system reports.
Normal flow of events:	1. User requests reports. 2. System sends the report to the user.
Sub-flows:	1. System fetches data from the UserLog. 2. System fetches data from the DeviceLog. 3. System generates reports.
Alternate/Exception flow:	No Alternate Flow.

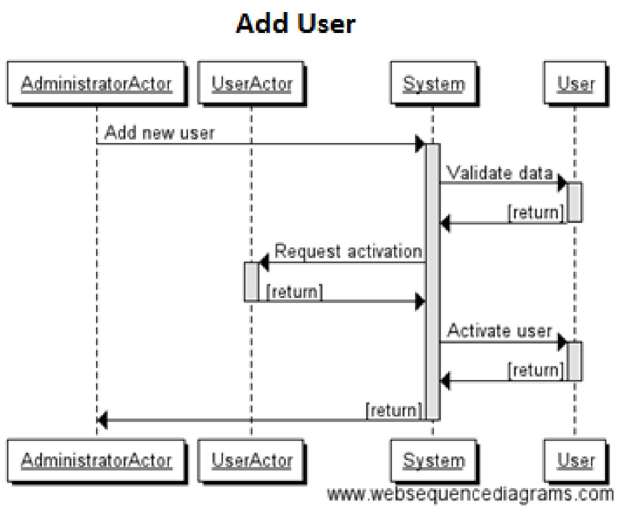
Use Case Name:	Add new user.
Primary Actor:	Administrator.
Brief Description:	Allows the primary user or administrator to add a new user.
Stakeholders:	Administrator, User.
Trigger:	When administrator adds a new user.
Normal flow of events:	1. Administrator requests to add a new user. 2. System returns a status response.
Sub-flows:	1. System validates data. 2. System adds user.
Alternate/Exception flow:	1. New User data is invalid. 2. User already exists.

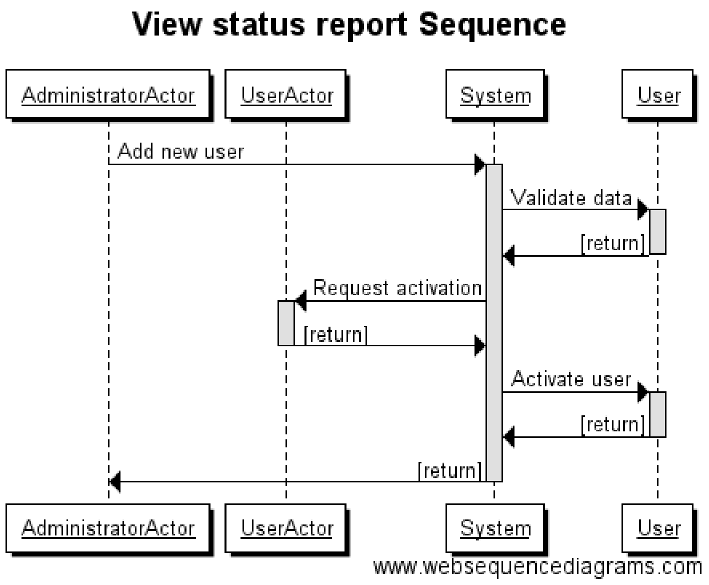
Use Case Name:	Trigger warning.
Primary Actor:	Sensor.
Brief Description:	Sensors sends a warning condition to system.
Stakeholders:	Sensor.
Trigger:	When there is an anomaly in the security system.
Normal flow of events:	1. Sensor sends warning. 2. System notifies administrator and users.
Sub-flows:	1. Validate device. 2. Check previous device log. 3. Generate critical alert.
Alternate/Exception flow:	No alternate flow.

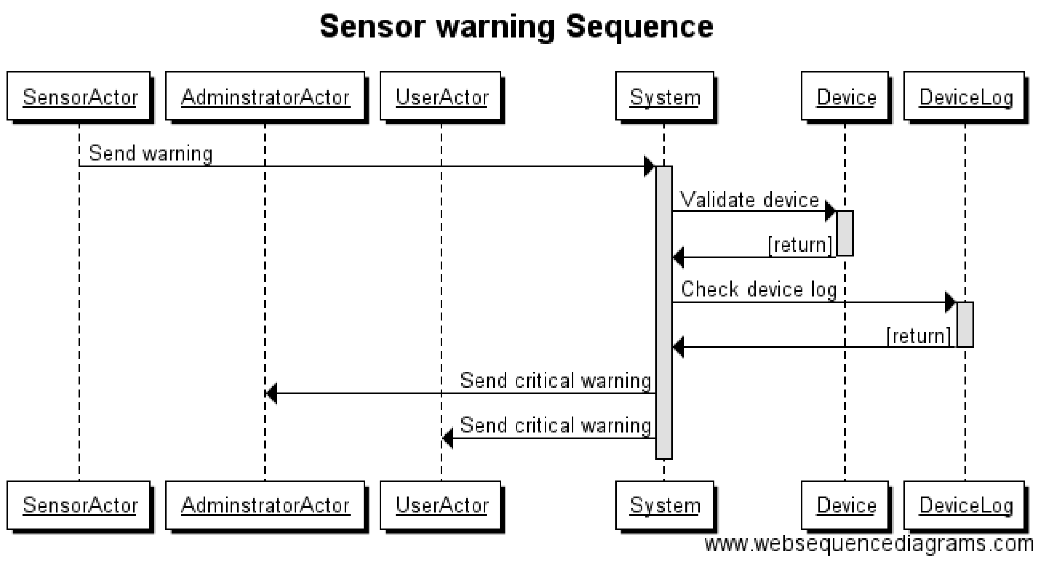
6. Dynamic Models






Add User
AdministratorActor
UserActor
System
User
Add new user
Validate data
[return]
Request activation
[return]
Activate user
[return]
[return]
www.websequencediagrams.com


View status report Sequence
AdministratorActor
UserActor
System
User
Add new user
Validate data
[return]
Request activation
[return]
Activate user
[return]
[return]
www.websequencediagrams.com


Sensor warning Sequence
SensorActor
AdminstratorActor
UserActor
System
Device
DeviceLog
Send warning
Validate device
[return]
Check device log
[return]
Send critical warning
Send critical warning
www.websequencediagrams.com

7. Design Documents

Software Design

Method Name: login	**Class Name:** User
Description of responsibilities: Implement the necessary behavior to login a registered user and create a session at a terminal.	
Arguments Received: loginPattern userID	**Data Type:** String String
Return Value: loginToken	**Data Type:** String
Message and Example: login(loginPattern, userID): String sessionToken = login(loginPattern, userID)	
Algorithm Specification:	

```
If loginPattern and userID is not null then
    If loginpattern and userID found in User table
        Return true
Return false
```

Method Name: viewStatusReport	**Class Name:** User
Description of responsibilities: Implement the necessary behavior to allow viewing of a status report on a terminal with a valid sessionToken.	
Arguments Received:	**Data Type:**
devices	String Array
time	String Array
Return Value:	**Data Type:**
report	Report Object
Message and Example: viewStatusReport (devices, time): Report Object report = viewStatusReport (devices, time)	

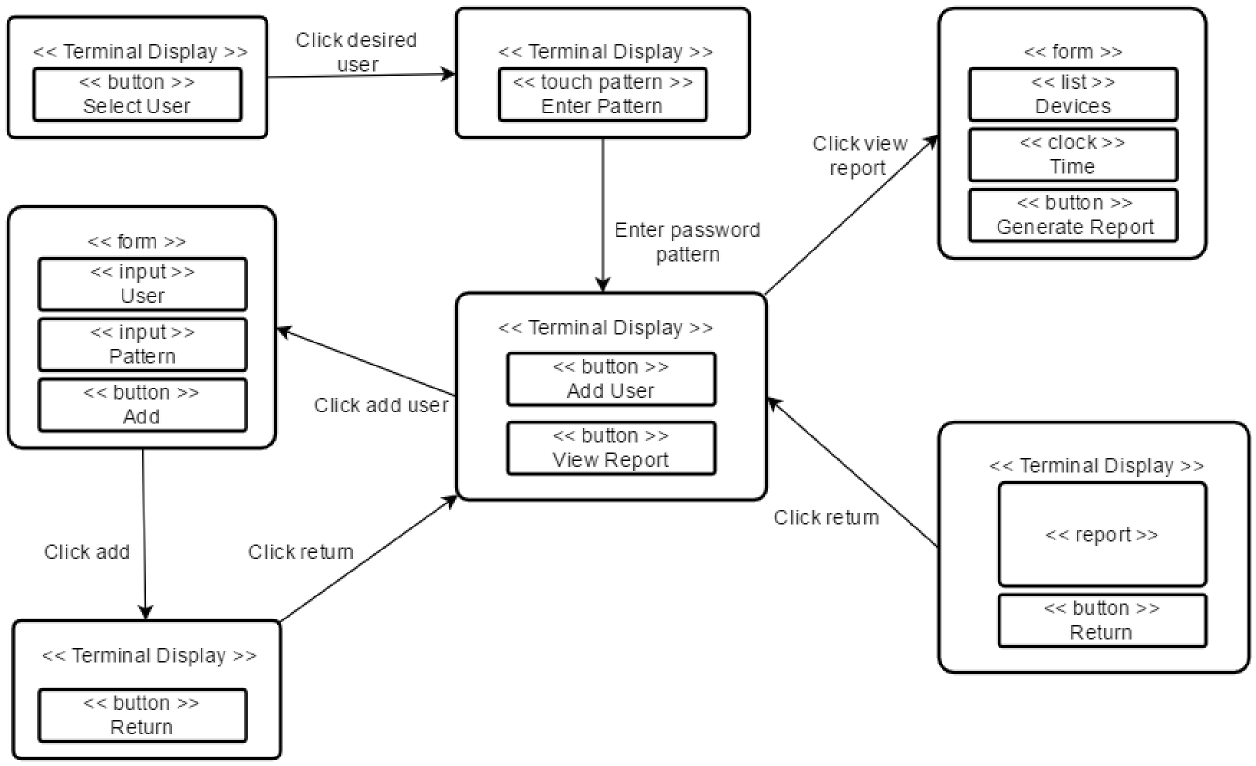
Algorithm Specification:

```
 If sessionToken is valid
    For each device in devices
        Retrieve device status based on time and append to report Object
    Return report
Return false
```

Method Name: addUser	**Class Name:** Admin
Description of responsibilities: Implement the necessary behavior to allow an authorized administrator to add a new user login into the system.	
Arguments Received: loginPattern userID	**Data Type:** String String
Return Value: registerStatus	**Data Type:** Boolean
Message and Example: addUser(loginPattern, userID): Boolean registerStatus = addUser(loginPattern, userID)	
Algorithm Specification:	

```
If loginPattern and userID is not null then
   If loginpattern and userID not found in User table
      Add new user into database
Return false
```

User Interface Design



8. Testing

Purpose

These test cases verify and evaluate the functionality related to the Smart Home service framework demonstrated by a device through the Smart Home interface.

The interface provides the centralized management capability to allow for a smart home client application to manage home appliances through the smart home server.

Scope

These test cases are designed to determine if a device conforms to the Smart Home interface specification. Successful completion of all test cases in this document does not guarantee that the tested device will interoperate with other devices.

Test Strategy

The most effective approach in this case would be a priority based testing method.

We shall Classify the defects as follows:

a. **High Priority –** Those which affect any aspects of testing in such a way as to cause a system crash or a severe misappropriation of system requirements such as but not limited to loss in reliability, security issues and functional limitations.

b. **Medium Priority –** Mildly severe to minor performance issues and functionality bugs that limit the system from functioning to its full potential.

c. **Low Priority –**System bugs which do not affect functionality and can include issues pertaining to but not limited to UI and design flaws.

Testing will be carried out in order from High Priority to Low Priority, however time and budget constraints will also be taken into account therefore some of the Low Priority issues may be fixed with a version update at a later time.

The testing will consist of a combination of manual as well as automation testing. Manual testing primarily in the latter stages with User Acceptance Testing to gain end user approval being a prime concern. Automation testing will help with regards to concerns pertinent to coding flaws, disruptions in system flow etc. All bugs and defects will be recorded in the HPQC so as to have a proper documentation and also give access to the stakeholders of the project testing status.

Requirements

The following are required in order to execute these test cases:

- A IOT ready device (the device under test or DUT) that implements the Smart Home interface according to the Smart Home service framework interface specification.
- A supported test device on which the test cases will run
- A Wi-Fi access point (referred to as the personal AP)

Preconditions

Before running these test cases, it is assumed that:

- The DUT is connected to the personal AP
- The test device is connected to the personal AP
- Note that the appliance to be controlled is simulated by the test device in this test case specification.

Test Case 1:

Objective

Verify that the Service Interface Version of the System App (the application supporting the Smart Home Service interface) is equal to 1.

Procedure

1. The test device listens for an About announcement from the application on the DUT.
2. After receiving an About announcement from the application, the test device joins a session with the application at the port specified in the received About announcement.
3. The test device calls the getVersion() property on the Centralized Management object.
4. The test device leaves the session.

Expected results

- The test device receives an About announcement from the application on the DUT.

  - The test device joins a session with the application at the port specified in the received About announcement.
  - The interface version returned from the getVersion property equals 1.
    - If not equal to 1, the test fails
    - If a bus exception occurs, the test fails.
    - Otherwise, the test case passes.

Test Case 2:

Objective

Verify that the test device can call the ApplianceRegistration() method to join the centralized management system.

Procedure

1. The test device listens for an About announcement from the application on the DUT.
2. After receiving an About announcement from the application, the test device joins a session with the application with the port specified in the received About announcement.
3. The test device calls the ApplianceRegistration() method on the DUT's centralized management object with the registration parameters (i.e., unique name and device Id).
4. The test device waits to receive the ReturnValue signal sent from the DUT.
5. The test device calls the Verification() method on the DUT's centralized management object with device Id to verify whether the test device registers on the DUT successfully.
6. The test device waits to receive the returned parameters (i.e., unique name and device Id) according to the device Id from the DUT.
7. The test device leaves the session.

Expected results

- The test device receives an About announcement from the application on the DUT.
- The test device joins a session with the application with the port specified in the received About announcement.
- The test device calls the ApplianceRegistration() method on the DUT's centralized management object to provide the registration parameters (i.e., unique name and device Id) and the method returns a OK response.
- The test device receives the ValidateCode included in the ReturnValue signal sent from the DUT.

- The test device calls the Verification() method on the DUT's centralized management object with device Id.
- The received parameters returned from the Verification() method on the DUT's centralized management object is the same as the test device's parameters (i.e., unique name and device Id) according to the device Id provided by the test device.

Test Case 3:

Objective

Verify that the test device can call the Execute() method to call the method of the appliance through the DUT. Note that the appliance to be controlled is simulated by the test device in this test case.

Procedure

1. The test device listens for an About announcement t from the application on the DUT.
2. After receiving an About announcement from the application, the test device joins a session with the application with the port specified in the received an About announcement.
3. The test device calls the Execute() method on the DUT's centralized management object with isReturn, deviceId, objectPath, interfaceName, methodName (i.e., ApplianceControl() method) and method arguments maintained with the validation test.
4. The test device waits for the DUT to call the applianceControl() method of the test device with interfaceName, methodName , arguments that is provided in the Execute() method call.
5. The test device sends the replied value of the applianceControl() method call to the DUT.
6. The test device waits to receive the ReturnValue signal sent from the DUT.
7. The test device leaves the session.

Expected results

- The test device receives an About announcement from the application on the DUT.
- The test device joins a session with the application with the port specified in the received an About announcement.
- The test device calls the Execute()method on the DUT's centralized management object with isReturn, deviceId, objectPath, interfaceName, methodName(i.e., ApplianceContol() method) and method arguments maintained with the validation test.
- The test device receives the method call from the DUT to call the applianceControl() method of the test device with interface name, method name, and method arguments that is provided in the Execute() method call.
- The test device sends the replied value of the applianceControl() method call to the DUT.

- The test device receives the replied value included in the ReturnValue signal sent from the DUT.