System Architecture

for

Machine Learning Smart Home

Prepared By:

Matthew Maynes

Cameron Blanchard

Peter Mark

Jeremy Dunsmore

Table of Contents

1. Introduction

1.2 Audience

This document describes the high level architecture of the Aria system. It details the major components within the system as well as their deployment artifacts. This document is intended for a technical audience to understand the organization, deployment and internals of the Aria system.

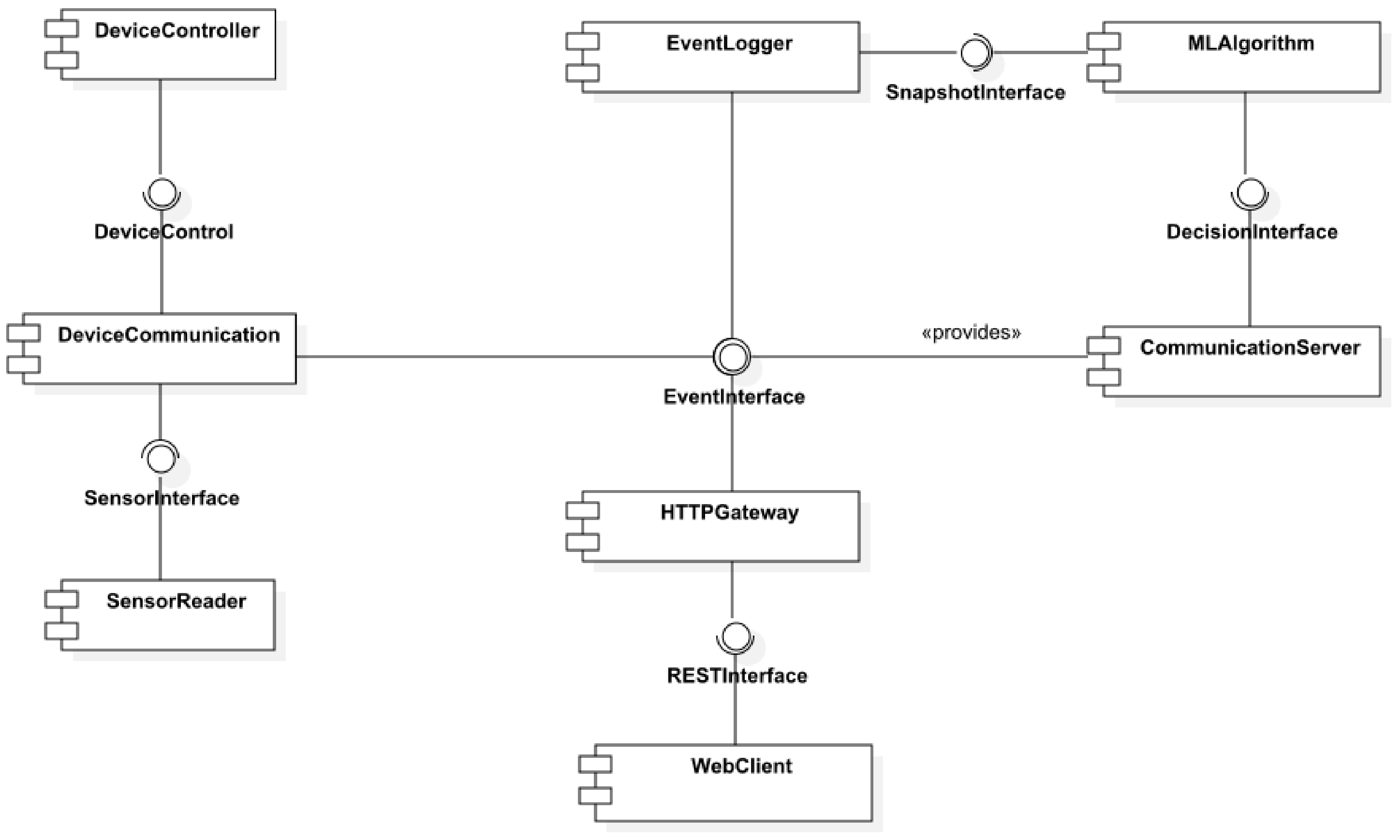
1.3 Background

The automated real-time interactive architecture (ARIA) system provides software for a machine learning smart home. The Aria system allows multiple smart devices to connect to a central hub to create a smart home. The system observes a user in a home environment and uses historical decisions to predict future actions.

This complex system is built upon several different subsystems that work together within a single hub. The hub communicates to several different devices over different protocols that are all synchronized with a custom Aria messaging protocol. This document outlines the details of these subsystems, interactions and the Aria messaging protocol.

2. System Components

2.1 Component Organization



2.2 Component Descriptions

Event Logger

The event logger is responsible for listening to all event signals and record them in a central database. This data will be accessible by the remote client as well as the ML algorithm component. The events recorded will be used to make decisions about actions to carry out. The event logger will consume the event interface that is provided by the communication server and will simply observe all data.

Event Interface

The event interface is the protocol that will be used to send event within the smart learning system. This interface will define the structure of data that will be sent from smart devices to the communication server as well as any other listening parties. Events that are sent from third party devices will use a different protocol to communication to the central hub. This protocol is designed for custom built devices.

ML Algorithm

This algorithm is responsible for reading events, generating a model of interactions and then making decisions about actions to perform. The implementation of this algorithm can be customized as long as it provides the decision interface. The ML algorithm will receive data from the event logger using snapshots of data. These data snapshots will reduce the amount of information that the ML algorithm needs to process by removing redundant information.

Communication Server

The communication server is responsible for routing all messages through the smart home system. The server has a cache of all connected devices and must store any related settings for each. Events and messages are routed through the communication server using the event interface. The communication server is also responsible for sending messages to other third party protocols. The server should provide a mechanism for installing plugins for other third party devices that are not natively supported.

HTTP Gateway

The HTTP gateway is responsible for supporting the web client interface. It must serve the web client's requests over a REST interface and translate them to the internal event interface used by the communication server. The gateway must be able to support multiple web clients.

Sensor Reader

The Sensor Reader is responsible for listening to a sensor and providing events when sensor data changes. The Sensor Reader will provide the Sensor Interface.

Device Controller

The Device Controller is responsible for controlling the physical device. It consumes events and modifies the output of the device accordingly. The Device Controller provides the Device Control Interface.

Device Control

This interface allows the device communication to pass events to the Device controller.

2.3 Component Interfaces

Decision Interface

The decision interface allows the ML algorithm to enter commands into the communication server which will be translated into events for the system. This interface must be provided by the ML algorithm and will be consumed by the communication server.

Snapshot Interface

This interface allows the ML algorithm to receive a subset of the data from the event logger. The snapshots remove any duplicate information before sending it to the ML algorithm for processing. This reduces the amount of data that needs to be sent as well as the amount of data that needs to be processed by the algorithm.

REST Interface

The REST interface allows web communication from web clients to the HTTP gateway. This representational state transfer protocol must allow the web client to carry out interactions with the various devices connected in the system as well as view the current system state. This interface must provide both a monitoring and controlling the system.

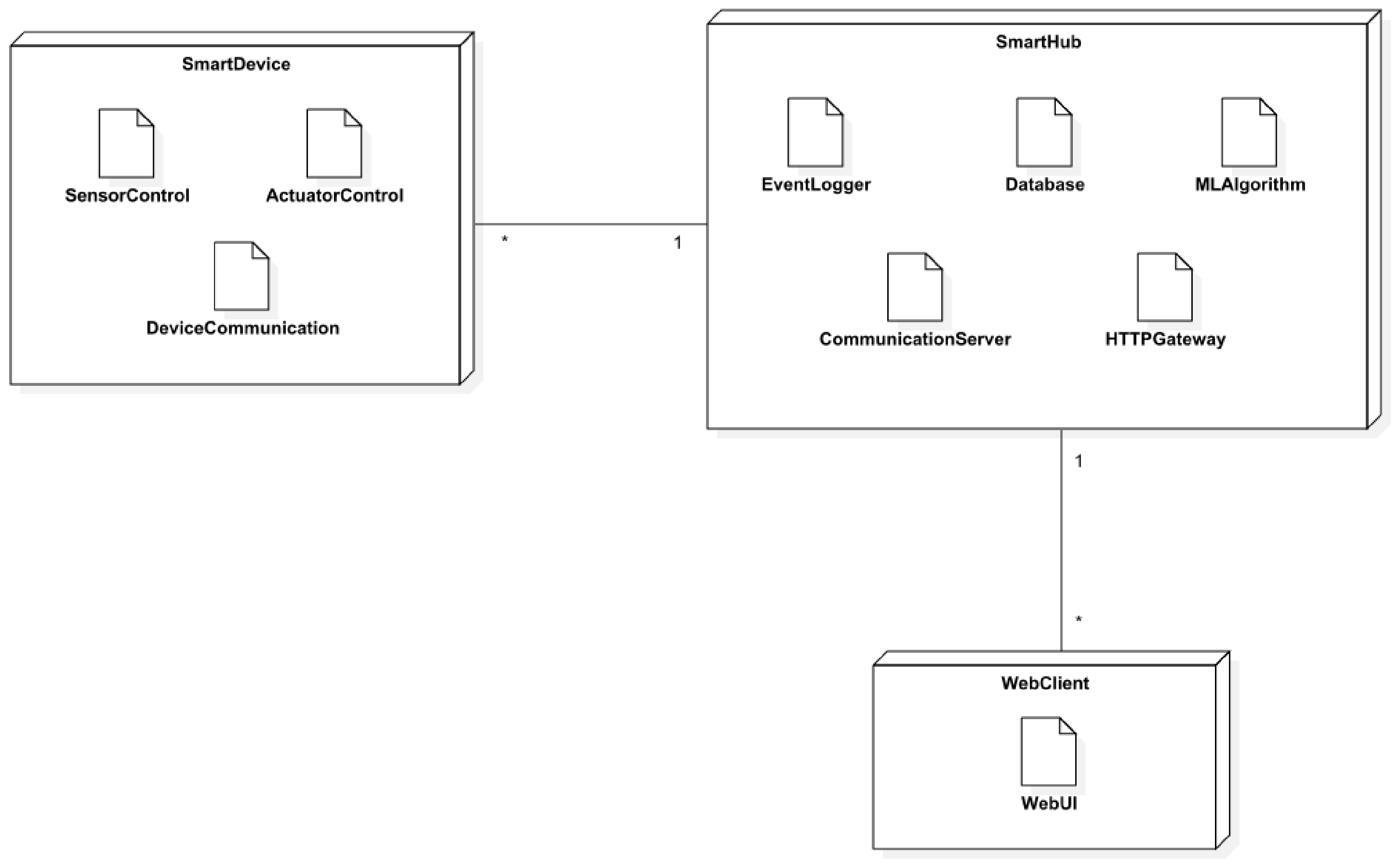
Device Communication

Device Communication is responsible for passing events between the device and the Communication Server. The Device Communication consumes the Device Control, Sensor Interface and the Event Interface.

Sensor Interface

This interface allows the Device Communication to read events from sensors and pass them on to the communication server. This interface is provided by the Sensor Reader and consumed by the Device Communication.

3. Deployment



Smart Hub

The central point of control of this smart home system is the smart hub. This hub is the central point of communication for all devices in the smart home system. It houses all of the data storage for events in the system and makes decisions using a smart learning algorithm. The hub will provide a minimal hardware interface for starting the system and changing the system state from training to normal to standby. The smart hub needs to be connected to a internet access point in order for it to serve the web interface to a client's computing device.

Smart Device

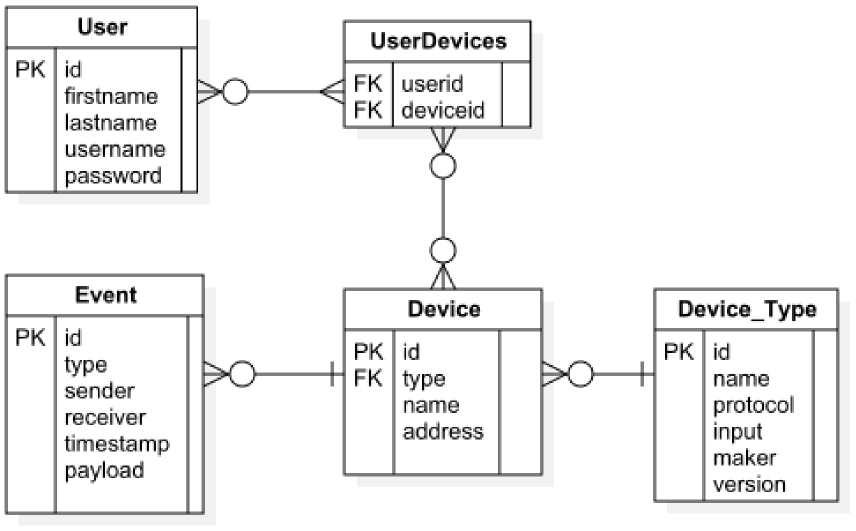
In this diagram, smart device refers to any smart device in the system. This could be a custom built device or a third party device. There will be many of these devices within the system all communicating to the central smart hub.

Web Client

The web client is the end user's browser and will present a remote interface for controlling the smart hub as well as all devices that are connected to the system. The web interface must be able to render on various industry standard browsers (Chrome, Firefox, IE, Safari).

4. Database

4.1 Database Entity-Relationship



4.2 Database Tables

Devices

The individual devices connected to the system

Device Types

The different types of possible devices, (WeMo switch,,Hue lights, etc...)

User

A person that can add and control devices

User Devices

The devices a user has access to

Event

A snapshot of a device state at a certain time

5. Device Protocol

This is a two level protocol. Messages are sent from the central server to a translation layer that converts the message into the device specific format. The intermediate layer will be consistent for all devices. The device handlers will need to support the basic API layer that is consistent for all devices but can implement it differently for any device.

The first level of the protocol is defined as the interaction between the communication server and the device handlers. This protocol is called the common communication protocol (CCP). The second level of the protocol is device specific and is the responsibility of the device handler. This layer of the protocol is termed the device communication protocol (DCP).

5.1 Common Communication Interface

This interface communicates directly to the communication hub. It must know how to convert between the CCP and DCP. The following methods must be defined.

Setup

This method will be required to setup the device handler.

```
setup (listener: Listener)
```

Discover

```
discover ()
```

Send

Send is responsible for the reliable transmission of data between the central hub and the desired smart device.

```
send (deivce : Device, message :  Message)
```

Teardown

```
teardown ()
```

Notifications

New Device

```
discovered (device: Device)
```

Message

```
message (device: Device, message: Message)
```

5.2 Device Communication Protocol

Messages are passed using the general message structure and are byte encoded. The payload of the message

```
+---------+---------+----------+-------------+--------------+
|  type   |   size  |  sender  | destination |   payload    |
+---------+---------+----------+-------------+--------------+
| 1 byte  | 4 bytes | 16 bytes |  16 bytes   | 'size' bytes |
+---------+---------+----------+-------------+--------------+
```

Message Protocol

Messages have to be sent between all components in the smart home system. The smart home system uses UDP to send messages. The following is the encoding structure for all messages sent in the system. It is assumed that all data in the payload field is JSON encoded unless the message type indicates otherwise.

General Message Structure

```
+---------+---------+----------+-------------+--------------+
|  type   |   size  |  sender  | destination |   payload    |
+---------+---------+----------+-------------+--------------+
| 1 byte  | 4 bytes | 16 bytes |  16 bytes   | 'size' bytes |
+---------+---------+----------+-------------+--------------+
```

Message Types

Name	Type	Value
Error	ERR	0x00
Discover	DISC	0x01
Request	REQ	0x02
Event	EVT	0x03
Acknowledge	ACK	0x04

Device Status

Return the device status and any properties associated with that device

Request

```
{
    "action" : "status"

}
```

Response

```
{
    "active" : "true|false"
    // ...
}
```

Device Information

Request

```
{
    "action" : "about"

}
```

Response

```
{
    "maker"     : "string",
    "protocol"  : "string",
    "version"   : "number",
    "name"      : "name"
}
```

Configure Device

Request

```
{
    "action" : "configure"
    // Fields that need to be configured
}
```

Field	Description
Display Name	Name displayed for device

Response

Response is status of update. True indicates success, false indicates failure

```
{
    "status" : "true|false",
    "error"  : "error message"
}
```

6. REST API

The HTTP gateway component exposes some REST endpoints. The gateway allows system components which can only communicate over HTTP to interact with any device in the system using a datagram protocol (see section 5 - Device Protocol).

REST API Design Goals:

- In order to ensure the system is extensible and to ease testing, The REST API should not change when new capabilities are added to the communication hub. The REST API strictly translates messages from HTTP to the Common Communication Protocol, with no interpretation of the requests and responses.

`GET /system/state`

Get information about the current state of the automation system Returns a JSON object.

`POST /request`

Send a request to the communication hub

Content-Type header should be `application/json` The body of the request should contain a JSON object. The object will be forwarded to the communication hub as the payload of a type 3 message.

`POST /devices/<id>/request`

Send a request to a a device in the automation network. Content-Type header should be `application/json` The body of the request should contain aa JSON object. The object will be forwarded to the device identified by as a type 3 message.

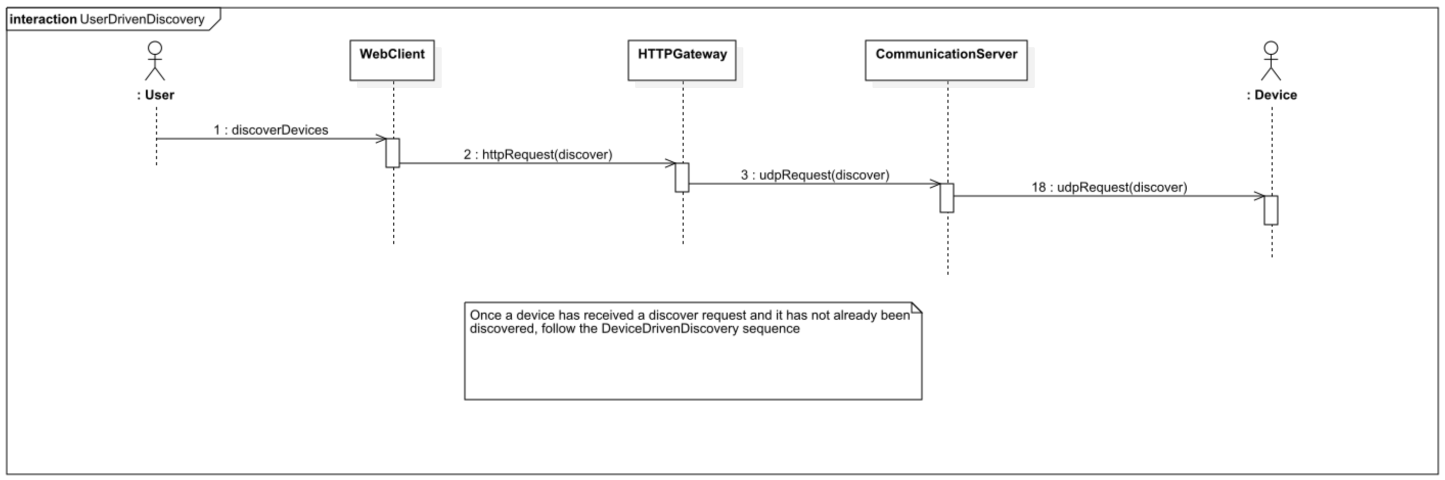
7. Discovery

7.1 Overview

In order to add a device to the central exchange hub registry, the device must be added through the discovery sequence. Until the device has been added, no communication can take place. Device discovery can be initiated from external devices or from the hub itself. This section outlines the discovery sequence in both scenarios.

7.2 User Driven Discovery

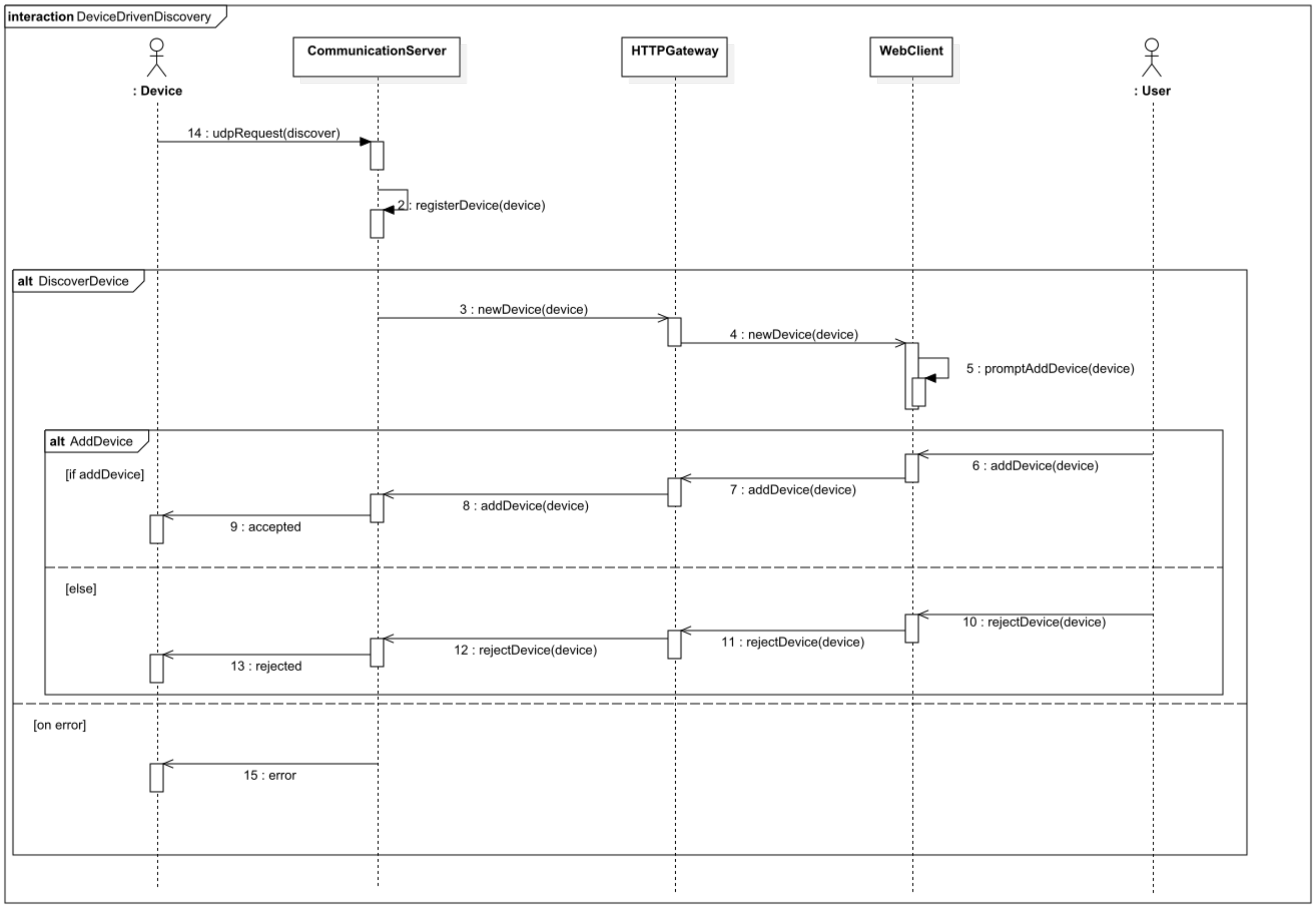
User initiated discovery begins with the user requesting the server to send a discovery request. The message then propagates through the system to the communication server which sends a broadcast discovery message to find new devices. If a device receives a request to be discovered then it begins the device driven discovery sequence



interaction UserDrivenDiscovery
: User
WebClient
HTTPGateway
CommunicationServer
: Device
1 : discoverDevices
2 : httpRequest(discover)
3 : udpRequest(discover)
18 : udpRequest(discover)
Once a device has received a discover request and it has not already been discovered, follow the DeviceDrivenDiscovery sequence

7.3 Device Driven Discovery

Device initiated discovery begins with an initial discovery request sent from the device. The hub will receive a request for the discovery with device details. If the device has valid information then the request is propagated to the web client. Once the web client receives a notification that a new device is available, it prompts to user to add the device. The user can then choose to accept or reject the device which will send a message to the device.


interaction DeviceDrivenDiscovery
: Device
CommunicationServer
HTTPGateway
WebClient
: User
14 : udpRequest(discover)
2 : registerDevice(device)
alt DiscoverDevice
3 : newDevice(device)
4 : newDevice(device)
5 : promptAddDevice(device)
alt AddDevice
[if addDevice]
6 : addDevice(device)
7 : addDevice(device)
8 : addDevice(device)
9 : accepted
[else]
10 : rejectDevice(device)
11 : rejectDevice(device)
12 : rejectDevice(device)
13 : rejected
[on error]
15 : error

8. Requests

8.1 Introduction

The purpose of this section is to provide an understanding of the interactions between an end user and the system for different types of requests.

8.2 Web Client

User

The User actor represents an end user of the system, which is the Web Client in this case. The user initiates all requests in the use case diagram for this system.

System Hub

The SystemHub actor represents the central hub of the ARIA system. The system hub is in charge of relaying information of the system to the User through the Web Client.

Device

The Device actor represents any device in the ARIA system. A device is responsible for relaying its specific information to the User through the Web Client.

List Devices

Return a list of all devices currently known by the system to the User.

View System Events

Return a log of recent events that have occurred in the system. An event is when the state of a device changes.

Get System Status

Return the status of the system to the user. This includes information such as what mode is the system currently operating in, current version number, number of connected devices, etc.

Set System Mode

Set the mode of the system. The different modes are: Standy(0x00), Normal(0x01), and Learning(0x02)

Add New Device

Add a new device to the system. This allows the end user to connect a new device to be controlled by the system and contribute to the machine learning.

Remove Device

Remove a device from the system. This allows a user to remove a device and its information, no longer allowing it to be controlled by the system. If added back to the system, there will be no stored information on it.

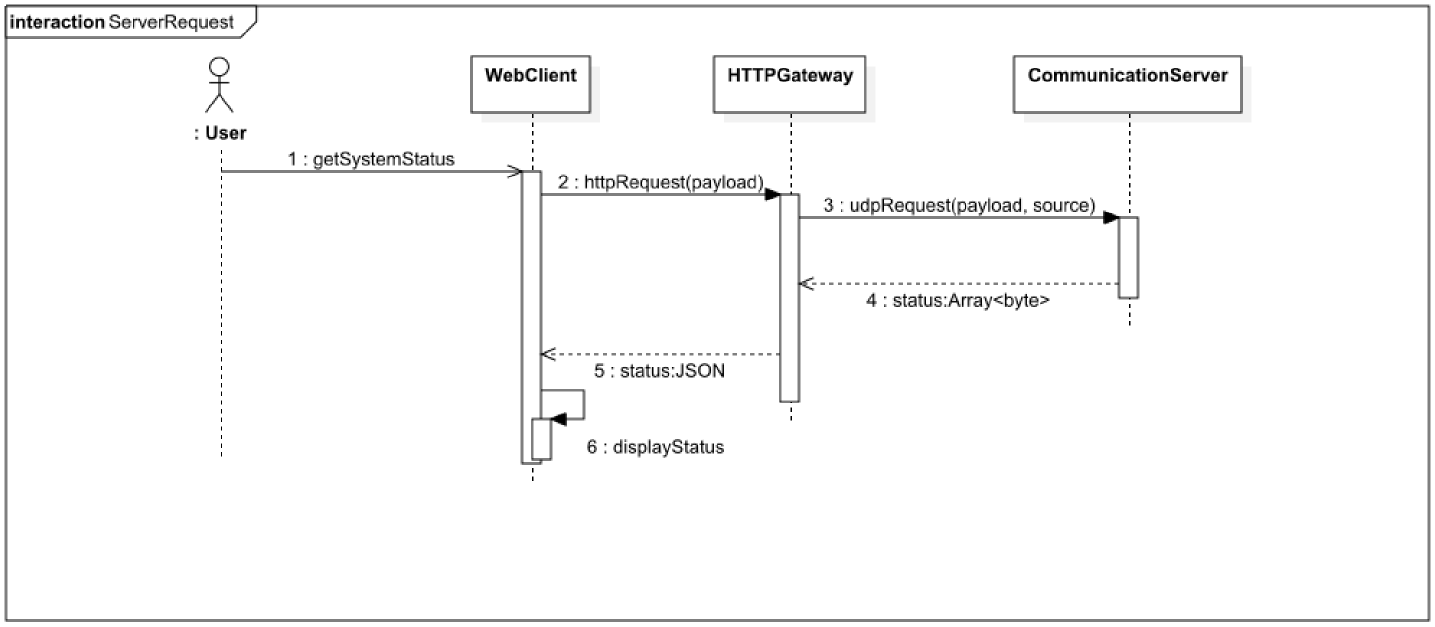
Get Device State

Return the current state of a device to a user. The state information could include if it is on, off, and other values depending on the device.

Set Device State

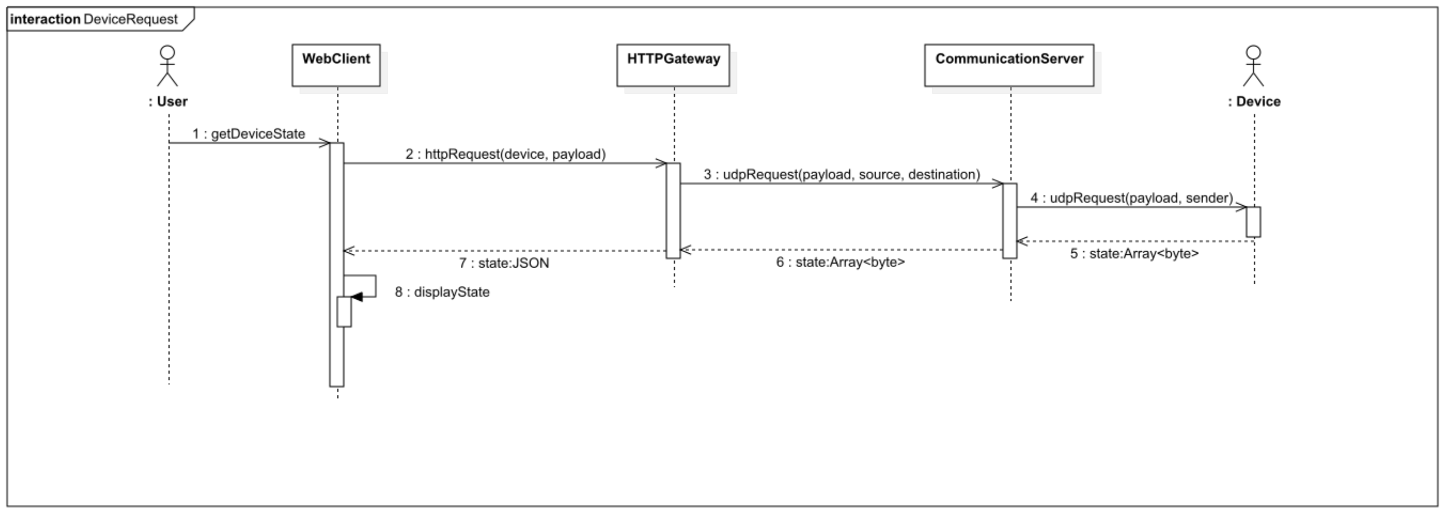
Set the current state of a device. The state information could include if it is on, off, and other values depending on the device.

8.3 Server Request



The Server Request diagram shows the workflow that occurs when an end user wants to retrieve information about the state of the communication server through the web client.

8.4 Device Request



The Device Request diagram shows the workflow that occurs when an end user wants to retrieve inform about a specific device in the system through the web client.