AIR-IMAG

POLYTECH GRENOBLE

Anna BRUEL

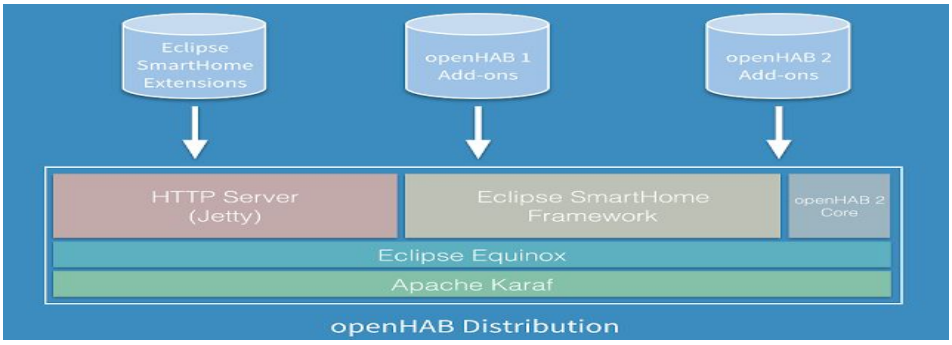
Cenyo MEDEWOU

Yacine NDIAYE

UPnPOpenHAB Project

Overview

The objective of this project was to be able to display video streams from cameras supporting the upnp technology and also to be able to remotely control them via OpenHAB which is a leading open source smart home solution.
To reach this goal, we had to understand the openHab interface and the Eclipse Smarthome framework.
The first step required to understand how the Eclipse Smarthome and OpenHab are related each other.
Since Eclipse SmartHome is only a framework to build smart home solutions, our project is the perfect example of what can be done with Eclipse SmartHome and OpenHab.
Of course, we worked with openHab 1 at the first place to understand the whole principles of the bindings and textual configurations. Then, we moved from OpenHab 1 to OpenHab 2, which offers more comfort to the user and a great ease of use.
The REST API of Eclipse Smarthome helped us a lot to access information about the cameras we discovered. In fact, it can give us access to the things (cameras) that we add in the Paper UI.
The goal of this report is to present the work we did with openHab and Eclipse Smarthome, and how to use the solution that we propose.



The picture above describes the main components of the openHab 2 Distribution and relations

between them.

Table of contents

Material Presentation

For this project, we have used 2 D-LINK IP cameras that support UpNP forwarding and several network protocols such as IPV4.

The first one is the D-LINK DCS-932L

It is a DAY/NIGHT camera that offers a MJPEG format video compression and JPEG still images.

It also includes different video resolutions:

- 640 x 480
- 320 x 240
- 160 x 112

The second one is the D-LINK PTZ DCS-5222L

It is also a DAY/NIGHT camera that offers a MJPEG format video compression, JPEG still images and pan/tilt functionalities. This functions will help us to test the remote control function of our project.

It includes different video resolutions:

- 1280 x 720
- 800 x 448
- 640 x 360
- 480 x 272
- 320 x 176

Software Presentation

Operating System: In this project, the team was using the Linux Operating System, meaning that all installations, configurations and tests have been done according to this OS.

IDE: Eclipse SmartHome is flexible framework that helped us to develop a new binding to discover the UpNP cameras located in a network. We also continued to use the IDE to easily compile and run the modifications we made in the ESH Extensions project that basically contains the code of the UIs.

Home Automation Platform: OpenHab 1 and OpenHab 2

Programming languages: mainly Java and also Javascript

Also, some other technologies have been used such as Felix, Maven for a first approach of discovering upnp devices with OpenHab. Then, we moved to Openhab2 and used Eclipse SmartHome that has already fully integrated Maven and other utilities. See below the detailed information about the technologies we used.

- UPnP (Universal Plug and Play) is a set of networking protocols that permits networked devices, such as personal computers, printers, Internet gateways, Wi-Fi access points and mobile devices to seamlessly discover each other's presence on the network and establish functional network services for data sharing, communications, and entertainment.
- iPOJO is a service component runtime aiming to simplify OSGi application development. It natively supports all the dynamism of OSGi. Based on the concept of POJO, application logic is developed easily. Non-functional properties are just injected in the component at runtime.
- OpenHab is an Open Source project that aims to allow control of equipments in a network by catching events. It also offers a nice user interface which is easy to configure.
- OpenHab2 is is an improvement of OpenHab 1 that offers more features.
- Felix-Apache Framework is an open source implementation of the OSGi Release 5 core framework specification.

- Eclipse Smarthome is a platform that allows the integration of different systems, protocols or standards and that provide a uniform way of user interaction and higher level services.

Project Description

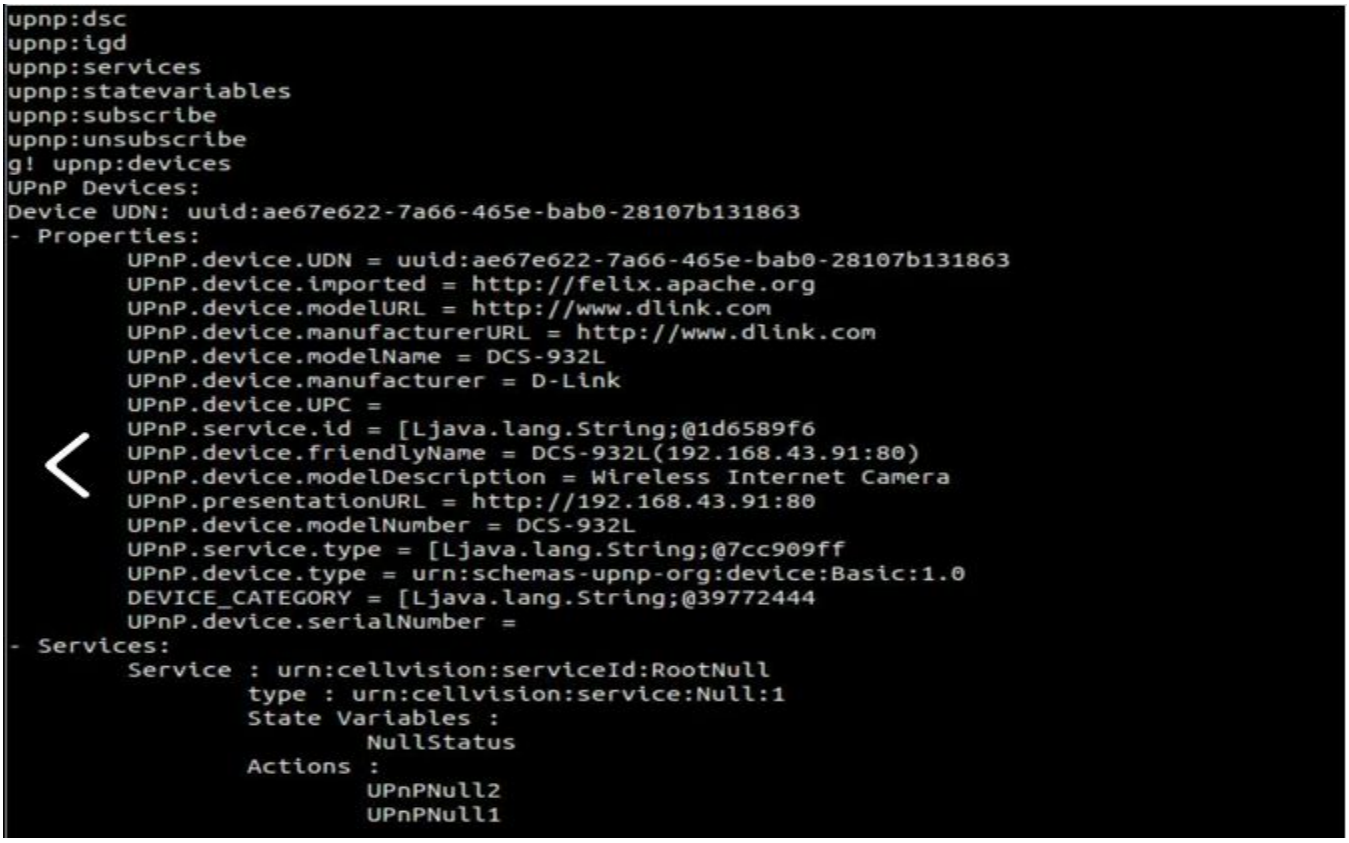
The first thing we did was to choose a namespace where to develop the new binding. The one we chose is org.eclipse.smarthome so we can directly contribute to the ESH project and permit the project to be used both for OpenHAB and other smarthome solutions.

1. First steps with Felix

In this part, we tried to automatically detect UpNP devices in a network, so we can have a preview of how the discovering works and see if the cameras can be discovered. We used Felix Apache and iPojo. With some dependency injection, we were able to add services and detect the cameras. This was our first unit tests that worked well.

Here we can see one of the cameras present in the network, in fact the Felix tester allowed us to test if our cameras support the UpNP technology.

The screenshot below are the results of commands that we developed using an osgi bundle to discover cameras and get some information about them such as the IP address, the UiD of the camera..



The first lines describe the available commands that we developed and here we tried to run the upnp:devices command. Here the command shows all the information about the camera, the properties and the services (this camera does not have any service).

2. The UPnP camera binding: `org.openhab.binding.upnpcamera`

In order to automatically discover the cameras, we provide a binding that can detect upnp cameras that are present in a network with the Eclipse SmartHome Binding Skeleton. This binding is the basis and represents a main part of our project and we will use it in the following to fully integrate the control of the camera in the OpenHab interface.

3. Customization of the User Interface

The video widget that is available on OpenHab cannot support the cgi format of the video streams. Therefore, we had to use the webview widget and customize it.

-The new webview for the binding

Each widget on OpenHab has an associated snippet written in html 5. With a specified render, we can access the information provided in the configured widget used in the sitemap.
In order to customize the webview that we use in the interface, we had to edit the webview snippet and the associated renderer. In the webview html file, we no longer take the url from the sitemap but we directly load the html file that we created for the cameras management with the src attribute of the iframe tag.
Informations related to a camera are sent through the url to the web page. We could then get the informations using a JavaScript function.

-The webview renderer

We have added some other classes to collect the discovering result of the cameras via the REST API with http requests. Then, we had to parse the Json data received from the rest API to load it in a new data structure that we could use after in the renderer.

-New packages

For a better namespace management, 2 packages have been added to the Ui:

```
Package
org.eclipse.smarthome.ui.classic.rest.render

And

package
org.eclipse.smarthome.ui.classic.rest.database
```

- The first package contains all the accessory classes that we need to correctly communicate with the cameras and the html file of the webview.

The `RestRenderer class:` It mainly provides the list of the cameras (with only the useful information) discovered by the upnpcamera binding. Here is where the data parsing of the Json result from an http request (Rest api) is made.

`-The UrlBuilder class:` The aim of this class is to check if a discovered camera is registered in some properties file (see the properties file section), and then build the right URL according to the information given by the file and the Ip address from the RestRenderer result.

`-Finally, we have the htmlBuilder class:` It is used to replace the values of the cameras in the html template file.

- The second package contains the `CameraGetPropertyValues class:` It is a class that provides all the camera information in the properties file in a ArrayList.

4. The properties file

A new resources folder containing the camera.properties file has been added

org.eclipse.smarthome.ui.classic/resources/**camera.properties**

```
# Syntaxe:
model|video_url|image_url|pan_url|tilt_url
                                                    --put null if an url is missing
```

The properties file adds a new security layer. In fact, the idea is to only display the cameras in the properties on the OpenHab interface.

5. Modification of the manifest file

In order to add the json library for the parsing, we needed to edit the manifest.mf file and add the bundle project to the classpath. So we have just added these lines in the file.

```
Export-Package: org.eclipse.smarthome.ui.classic.render
```

```
Bundle-ClassPath: ., lib/javax.json-1.0.jar
```

- We put your jar file in the file system of the project (lib/javax.json-1.0.jar) and added it to the bin.includes section of the build.properties file.
- To compile the binding in Eclipse, we also had to add the library to your .classpath as well.

Conclusion

About the main goals of the project, we can say that we have achieved them and working with OpenHab and Eclipse Smarthome was a great experience in a context of conviviality and sharing.
The general idea of the community is to publicly share ideas and make the openHab/Eclipse SmartHome a great tool to improve home automation.
For further information, go check the github of our project on https://github.com/openHab-UPnP

Useful links:

https://github.com/openHab-UPnP

- node-upnp-client : A pure Javascript upnp client
- OpenHAB
    - See the OpenHAB github
        - UPnP related sources
        - VideoRenderer
- UPnP
- Digital Security Camera V 1.0
- OSGi UPnP Base Driver
- [Javadoc OSGi UPnP Driver]
- Simple UPnP Command for the Gogo shell
- projet d'extension XBMC : source de l'addon DLink
- iSpyConnect (sources on GitHub)
- Kai Kruzer / OPenhab 1 to OpenHab 2
- Eclipse Smarthome for Developers
- Take a look at binding dependencies !
- Developing new binding for openhab2
- Set up your IDE with ESH!