Vigié benjamin (vigie_b)

Training period report

Final study internship report carried out in NDS France from 03/19/2012 to 09/21/2012

PART One:

Part TWO

Part THREE

Part one

This part is an official company document for a person who will continue the project on which I worked. We assume that this person is a new employee. Through this document I quickly describe the company and the project team before focus on the project. Finally, I will give some advices and directions for he can really understand what we expect and how to develop it.

OpenMHA

RAPI overview and guidelines

Purpose of This Document

The goal of this document is to provide a simple and quick way to understand the RAPI project and to be able to continue work on it.

First of all, this document presents our company before focus on the project. After introduce the background by describe OpenMHA, it will detail the purpose of this document: the RAPI project.

After reading this document, you will be able to continue the project, improve its specification and use all requires tools to start your work.

Doc. Title:	OpenMHA - RAPI overview and guidelines
Version	1.0
Author	Vigié Benjamin

Background and presentations

NDS quick overview

About the company

NDS is a leading global provider of end-to-end software solutions for the pay television industry. We enable pay-TV operators to offer a complete viewing experience to their subscribers and ensuring that only paying viewers can view content.

Over one-third of the world's digital pay-TV households rely on our technologies to discover, navigate or interact with content. Our clients include many of the largest cable, satellite and broadband pay-TV operators, including Astro, Bharti, BSkyB, Canal Plus, China Central Television ("CCTV"), Cox, DIRECTV, Kabel Deutschland, Sky Deutschland, Sky Italia, TataSky, UPC (a unit of Liberty Global) and Vodafone. NDS is very active in emerging markets where the move from analog to digital is now occurring.

Pay-TV operators also want to remain the one-stop-shop for all of their subscribers' content needs. NDS gives them the technology to provide all types of content to all types of devices–keeping customers loyal while increasing revenue potential.

NDS main Products

We enable pay-TV operators to deliver a unique experience to their subscribers. Our solutions protect and deliver content, while rich, intuitive interfaces enable access to all types of content.

The main company's activity is to provide rich and intuitive software solutions. More of a simply deliver an end-to-end solutions to enjoying TV anywhere, anytime and on any device, the main asset is the content and service protection. NDS protects the digital content and the operator's service.

The key software products in our end-to-end solutions include:

- XTV™ Digital Video Recorder ("DVR") Software. Our DVR software enables pay-TV subscribers to record content as well as pause and rewind "live" TV. It has eliminated schedules and changed viewing patterns.

- NDS EPGs, including NDS Snowflake™, allow subscribers to easily and intuitively discover, navigate and interact with content and features provided by the platform operator on a variety of devices.

- VideoGuard® CA and VideoGuard Connect™ DRM ensure that only paying subscribers can access content on all types of devices. They protect the valuable digital content as well as the operator's service. Robust content protection gives operators access to premium content for a more compelling offering.

- MediaHighway® Set-top Box Software. Our middleware software, similar to an operating system, enables operation of set-top boxes and delivery of content over both broadcast and broadband networks as well as over-the-top to broadband connected set-top boxes. The RAPI project, target of this document is a part of the new version of this solution, called OpenMHA and described below.

France Office

Near by the UK corporate headquarter; the French office is the smaller in the group. 327 people works in Issy-les-Moulineaux, which represents around 8% of the total company employees.

Before be a part of NDS, the French branch was the *Canal Plus* tech division. Because of this legacy *Canal Plus* still remains one of the main French partners of the company. The RAPI project has been initiated for *Canal Plus*.

OpenMHA quick overview

The RAPI project is a part of a bigger project called OpenMHA. It's better to have a complete point of view of the global solution to fully understand the RAPI project.

The OpenMHA project

OpenMHA is the evolution of the NDS main product MediaHighway. OpenMHA, for Open MediaHighway Advance, focus on the use of a maximum of free and open source project. More reasons to this choice:

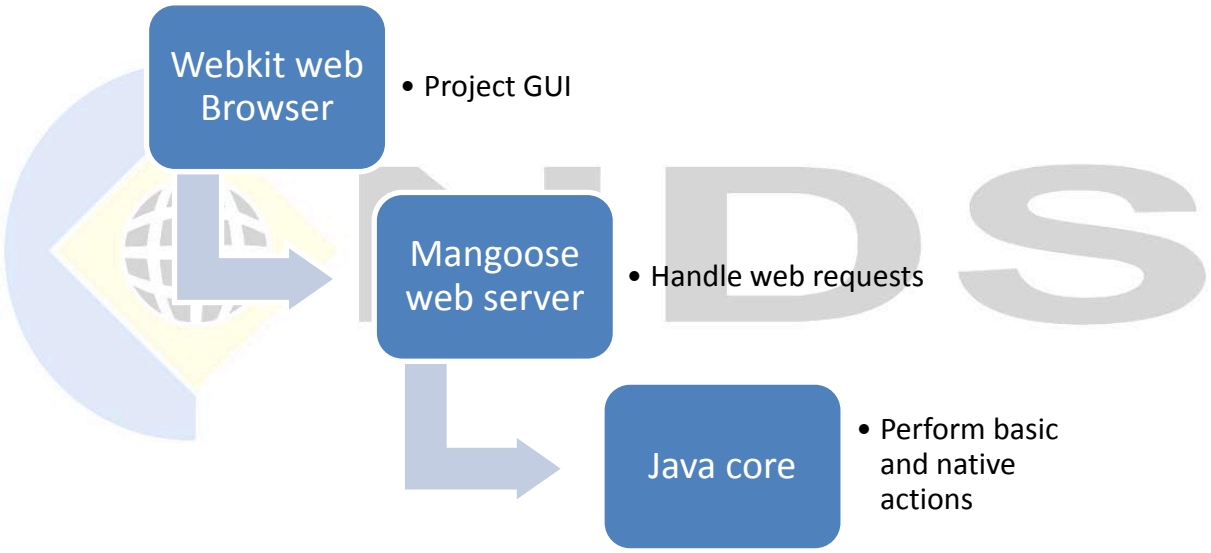
- Offer a cheaper product
- Be more respectful and compliant with standard
- Use the community to improve and accelerate the development
- Participate and improve standard, as QT, Webkit, DVB and so on.

OpenMHA's main idea is to combine a web browser and HTML5 as the main user interface. It's very important to notify that, because it's a major evolution in project architecture and the reason we develop the RAPI project.

Historically, the user interface was in JAVA, but it's a kind of old and complex technology. Pay-TV operators were obliged to deal with us for any application. It was very restrictive for them and painful for us to develop everything from A to Z. The solution chosen by OpenMHA was to provide an easy to code solution: with HTML5 and RAPI, pay-TV operators are able to develop their own TV applications easily and quickly.

OpenMHA architecture

We can basically divide OpenMHA in 3 parts:



- The Webkit web browser, the front-end part. JAVA user interface is replaced by web pages viewed in this browser. To interact with the hardware and to play or manage TV content, we use the RAPI, purpose of this document.

- The Mangoose server, a webserver developed in C. This server is a kind of bridge between the JAVA part and the web browser. Its role is to catch web request (indeed, the RAPI's ones) and to do some actions. For low-level interactions, it can call JAVA services.

- The JAVA core, a legacy of the old architecture. JAVA core provide all basics interactions with the hardware like setup the tuner, play video and audio streams, manage the front display and so on. The final goal of OpenMHA is to reduce this slow and resource-intensive part.

The RAPI project in OpenMHA

To provide the opportunity to develop a rich web application for the TV, NDS have to provide some tools to deal with the STB. That's the reason we develop the RAPI project. It's an essential component of the OpenMHA strategy because it's the bridge between our middleware and their applications.

We will develop in details the RAPI project and what it requires in the next part.

RAPI project

Detailed presentation

The OpenMHA REST API (better known as RAPI) offers an application API that allows developers to integrate and control end-users' STB interactions. Applications can browse available programs, detect what is being viewed, schedule programs to be recorded, and more.

The second screen devices like tablet, smartphone… are deeply modifying the TV viewing habits. As a consequence, the NDS customers (the TV operators) want to provide some access to their services to these popular CE devices (smart phone, tablet, connected TV, PC …).

It will provide a unified user experience on every user device. In using a remote UI solution, the operator keeps the users in the operator branded environment. The solution is very flexible too because it allows TV operators to easy manage or change the brand identity without a lot of code and big updates. Moreover the cost of application development is reduced as the same application code can be run on several devices.

The ultimate goal of the NDS RUI framework is to allow the operator to deploy application and services in a consistent way among every device in the home.

The targeted devices are beside the operator controlled STBs, all the CE devices that can be found in a modern home such as:

- Tablets and smart phones: iOS and Android
- PC, Mac
- Game Console : PS3, XBox …
- Connected TV featuring a web browser (mainly CE-HTML today but HTML5 to be supported in a near future).

The RUI framework is application engine agnostic. It can work with any application engine such as JVM, Flash, Unity3D.

The RUI APIs are specified in WebIDL language and as such several embodiments can be supported such as Java, JavaScript or C++.

However the preferred solution is an HTML5 / JS application engine. This solution seems to be the most future proof.

RUI Definition

A remote user interface is made of two elements the remote UI client and the remote UI server. There are different RUI technologies according to where the UI is actually rendered.

In the true RUI solutions, although the UI is obviously displayed by the client, the UI is rendered in the server side. Among these technologies we find: Active Video RDP, VNC, RVU…

There are other RUI solutions in which the UI is rendered in the client side. These RUI solutions are actually more distributed middleware services rather than remote UI sensu stricto, as the rendering engine is located in the client side. Among these solutions we have HTML/JS applications, Flash applications… The NDS RUI solution belongs to the second category as the preferred technology is HTML5/JS.

RAPI architecture

The NDS RUI Framework exports a set of APIs to provide the applications with digital TV functionalities.

These APIs are organized in clusters. Each cluster implements a functional category. The API contains the following clusters (non-exhaustive list):

- Framework cluster: The entry point of the API, to discover the available services.
- Content cluster: To browse, search and purchase content.
- Device cluster: To discover and to control the devices found in the Home Network.
- User profile cluster: To identify the current user and to set the user preferences.

In order to keep a readable document, you can find the diagram of each cluster in appendices.

There are two levels of API are exposed to application developers:

Dedicated High level API:

These APIs exposed high level functionalities to simplify the development of typical TV component like Grid, zapping, Recording, VOD...

Generic low level API:

These APIs are lower level. There are intended to be used by “advanced developers” to implement functionalities that go beyond the High Level APIs.

Folder hierarchy

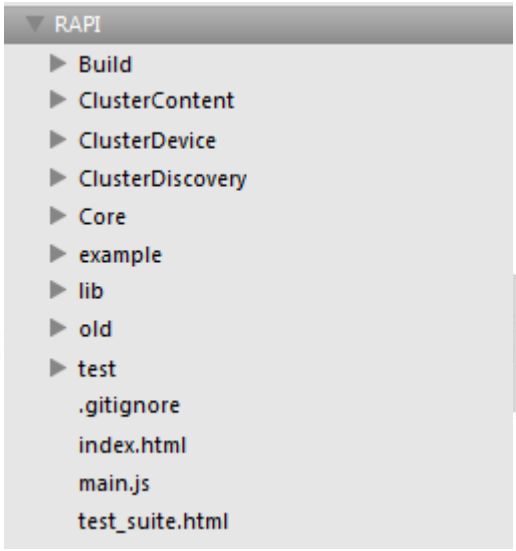


Figure 1 RAPI project folder hierarchy

- Build contains a script to launch a RAPI build. It will be detailed in the "Build" section.
- All the folders starting by Cluster contain the implementation of the different clusters.
- The Core folder contains the code shared across the clusters.
- The example folder contains some utilization examples made by Benoit Boyle.
- The old folder contains the implementation made by Benoit Boyle.
- The test folder contains the different unit tests.
- index.html permits to launch the RAPI
- test_suite permits to launch the unit tests

1. Require.js : Sand-boxing, dependancy managment and building

Official presentation

RequireJS loads plain JavaScript files as well as more defined modules. It is optimized for in-browser use, including in a Web Worker, but it can be used in other JavaScript environments, like Rhino and Node. It implements the Asynchronous Module API.

RequireJS uses plain script tags to load modules/files, so it should allow for easy debugging. It can be used simply to load existing JavaScript files, so you can add it to your existing project without having to re-write your JavaScript files.

RequireJS includes an optimization tool you can run as part of your packaging steps for deploying your code. The optimization tool can combine and minify your JavaScript files to allow for better performance.

Sandboxing and module

Require.js (http://requirejs.org) permits to define the application in module with the AMD syntax. You should read this excellent paper about the way to create module in JavaScript: http://addyosmani.com/writing-modular-js/

The Asynchronous Module Definition (AMD) API specifies a mechanism for defining modules such that the module and its dependencies can be asynchronously loaded. This is particularly well suited for the browser environment where synchronous loading of modules incurs performance, usability, debugging, and cross-domain access problems.

Basically, Require offers to main functions define() and require(). define() permits to create an AMD module and require permits to use an AMD module or a classic file.

All the detail of require.js usage are in http://requirejs.org. It's well documented and full of examples.

Dependency management

Each module has to declare its dependencies. This will assure that the files are loaded in the right order.

An AMD module will not create any variable in the global scope. It's very important because the global variables could be override by the application using the RAPI.

Furthermore, a variable that is set in the global scope cannot be garbage collected and will stay in memory for all the application life.

Currently, the RAPI write the entire interface object directly in the global scope. This is a problem and

should be corrected.

Require.js is currently in the version 1.0.7 and doesn't support "unload" feature. So, for the moment, a module that is loaded by require.js will stay in memory because the module cannot be garbage collected since Require.js keeps a reference of the entire module.

In the roadmap, Require.js 1.1 will add an unload feature. The current work is visible here: https://github.com/jrburke/requirejs/tree/165-undef.

This will permit to manage the memory very closely.

2. Unit testing for the RAPI

Mocha.js : establish the test suite structure

Mocha (http://visionmedia.github.com/mocha/) is a feature-rich JavaScript test framework running on node and the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

It's an excellent library that is highly flexible. You can use different grammar to write your test: TDD, BDD. In the RAPI project, the BDD style is preferred.

Here is a quick example to how describe a BDD test :

```
describe('Array', function(){
  describe('#indexOf()', function(){
    it('should return -1 when the value is not present', function(){
      [1,2,3].indexOf(5).should.equal(-1);
      [1,2,3].indexOf(0).should.equal(-1);
    })
  })
})
```

Note : For the CI, it will be possible to launch the tests on node.js. However, some changes are needed because there are some conflict with require.js and the require build-in function of Node.js.

Chai.js : contains the assert method

Chai.js (http://chaijs.com/) is a library that works very well with mocha.js. Chai.js will define multiple grammars to test a variable state. The syntaxes are “should”, “assert” or “expire”.

In the RAPI project we are using the “should” syntax that offers a very expressive code. For instance, we can write thinks like that:

```
v.should.be.an.instanceof(Array)
```

Sinon.js: mock and stub

Sinon.js (http://sinonjs.org/) is the last part of the combo. It handles the mocking and stubing of the object. It's actively used to mock the server response or to fake the xhr request.

Here is an example:

```
this.server.respondWith("GET", "/content/av-event?first=0&count=3",[200,
        { "Content-Type": "application/json" },
                window.JSON.stringify([
                    {
                "_interface" : "ITYPE_ID_AVEVENT"
                },
                {
                "_interface" : "ITYPE_ID_AVEVENT"
                }
            ])
      ]);
```

The RAPI project

Project snapshot

What we have to do

As say in first part, the RAPI project is attached to OpenMHA project. This project is still in development, but we have to provide some RAPI's functionalities to promote the project by showing how it's easy to develop applications.

We have split the development in several parts called "Scope", which involve a "vertical development": several clusters are require to achieve functionality. Actually we do a "horizontal development" by made a cluster from A to Z in order to have a complete brick of the project.

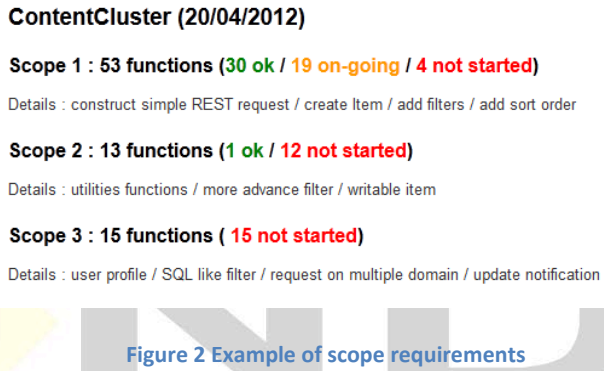


Figure 2 Example of scope requirements

Remember that you have to develop unitary tests and functionality tests when you are in charge of a cluster.

Unitary test are design to independently test a function of the cluster. They are made with the "Mocha.js test suit" (see part one, *RAPI's tools and requirements*). To add a test you can start by copying the test-squeleton.js file then add the new created file in test_suite.js.

Once again, the RAPI project is essential in the OpenMHA strategy.

What is already done

As it said previously, the work flow is focused on the cluster, not on the lots. However, the scope stay the reference for our work.

Here is a summary for each cluster:

- Discovery cluster: 100% complete. This cluster implements also an interface out of the specs who deals with the HTML5 Websocket. It is strongly recommended to carefully study this code because you will have to use it. Indeed, it's the official way to subscribe to server's events (including records events).

- Content cluster: almost finish. The only part remains is a special part focus about low-level filters. Decision was taken to keep this part alone for the moment because the server doesn't choose the way to treat data's. So Content Cluster doesn't require any work now

- Device Cluster: As the Content Cluster, the Device Cluster is considered as achieve for the moment, but in a near future you have to upgrade the IDevice interface. Indeed, this interface inherits form IServiceFinder. Today this inheritance is not implemented. It will be one of your tasks to implement it.

- Record Cluster: Last but not least, the Record Cluster is not yet implemented. It will be the main part of your future work.

What we expect

The most urgent task today is to implement the Record Cluster. Indeed, it is the last important point remains in the LOT 2. If we want to show a complete TV user experience, we have to demonstrate our ability to perform and manage recordings.

Once this cluster will be terminated, you should develop a demo application to demonstrate functionalities of the Record Cluster and its integration into the RAPI. The best way to do that is a program grid, but you are free to choose any other way to use records.

Requirement and advices

What you should do

First of all, it strongly recommended feeling comfortable with JSON format and REST protocol because you will manipulate a lot of data. Here is some interesting links which can help you to update your knowledge.

JSON format

http://tools.ietf.org/html/draft-zyp-json-schema-03

http://json-schema.org/

REST protocol

http://fr.wikipedia.org/wiki/Representational_State_Transfer

http://www.xml.com/lpt/a/2004/12/01/restful-web.html

http://www.xfront.com/REST-Web-Services.html

http://www.figer.com/publications/REST.htm

To begin on the RAPI project, I recommend staying focus on the Record Cluster. It's the last important part of the LOT 2 and it should be a good program entry point for someone who has to implement the API. I recommend starting with the IEventRecording (see specification diagrams in appendices). Don't hesitate to collaborate with Marc Mouille to fully understand what to do and what to send to the server.

Rapi's best practice

During your development, don't forget to always provide unitary test for each method you implement. It's very important because it's the only way for OpenMHA to test the project's progress.

As important as unitary test, you can develop some demo applications. Demo applications are HTML and JavaScript pages to demonstrate a real life utilization of the RAPI like a program grid or a zapper banner. You can find some demo applications in the RAPI_Implementation directory.

Be rigorous on your demo application because they can be the final client demo application of the RAPI.

For technically advices, keep in mind that you are on STB. It involves that you are on a platform with limited resources. Try to avoid many item creation / deletion which consume a lot of resources (read this excellent article about garbage collector and performance: http://buildnewgames.com/garbage-collector-friendly-code/).

Use Require JS at its maximum. It's a powerful tool to avoid global-scope variable declarations and create well-formatted objects. Another advantage is API implementation will be hidden by provide only interface objects.

Generally try to improve your code as far you can. Avoid at maximum string and array duplication, long class or array iterations, use of floating numbers. Try to use timestamp instead of Date() and so on.

In case of emergency

You can contact each project teammate describe before. But remember if you have any question regarding the specification; please contact Jean Brun (office phone 6732) or Severin Rataninko (65 34).

For implementations problems or if you not sure about a request parameter, it's better to deal with Marc Mouille (98 56). Indeed, he is in charge of the server side so don't hesitate!

Finally, for any JavaScript technical question you can contact Benoit Boyle (56 88), senior JavaScript developer and initiator of the RAPI. If there is something you want to improve or debug, you be sure he had the response you want.

Glossary

OpenMHA: Open Media Highway Advance, the new NDS project based on Media Highway Advance. The goal is to reduce the old and slow JAVA part by using open-source software.

RAPI: For REST API. It's an API project to allow a web browser to deal with the STB.

REST: Representational State Transfer. It's a communication protocol. It's a style of software architecture for distributed systems such as the World Wide Web. REST has emerged over the past few years as a predominant Web service design model.

JavaScript: JavaScript is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

HTML5: HTML5 is a markup language for structuring and presenting web content, and is a core technology of the Internet. It is the fifth revision of the HTML standard and is still under development.

CSS3: Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in HTML. CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts.

JAVA: Java is a set of several computer software products and specifications from Sun Microsystems that provide a system for developing application software and deploying it in a cross-platform computing environment.

Mangoose: Open source web server developed in C by Google.

Web server: Program who handle client request on a machine and give asked files (if allow). It use HTTP protocol.

QT: QT is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI), and also used for developing non-GUI programs such as command-line tools and consoles for servers.

Webkit: WebKit is a layout engine designed to allow web browsers to render web pages. It is also used as the basis for the experimental browser (likeAmazon Kindle ebook reader) as well as the default browser in the iOS, Android, BlackBerry Tablet and so on. The WebKit engine provides a set of classes to display web content in windows, and implements browser features such as following links when clicked by the user, managing a back-forward list, and managing a history of pages recently visited.

DVB: Digital Video Broadcasting (DVB) is a suite of internationally accepted open standards for digital television. DVB standards are maintained by the DVB Project, an international industry consortium with more than 270 members. Many aspects of DVB are patented, including elements of the MPEG video coding and audio coding.

STB: A set-top box (STB) is an information appliance device that generally contains a tuner and connects to a television set and an external source of signal, turning the source signal into content in a form that can then be displayed on the television screen or other display device. Set-top boxes can also enhance source signal quality.

API: An application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables. An API specification can take many forms, including an International Standard such as POSIX or vendor documentation such as the Microsoft Windows API.

Stream: In TV world, the term stream mean “TV signal”. It represents the image and sound data flow received by the STB.

Appendices

1. Content Cluster

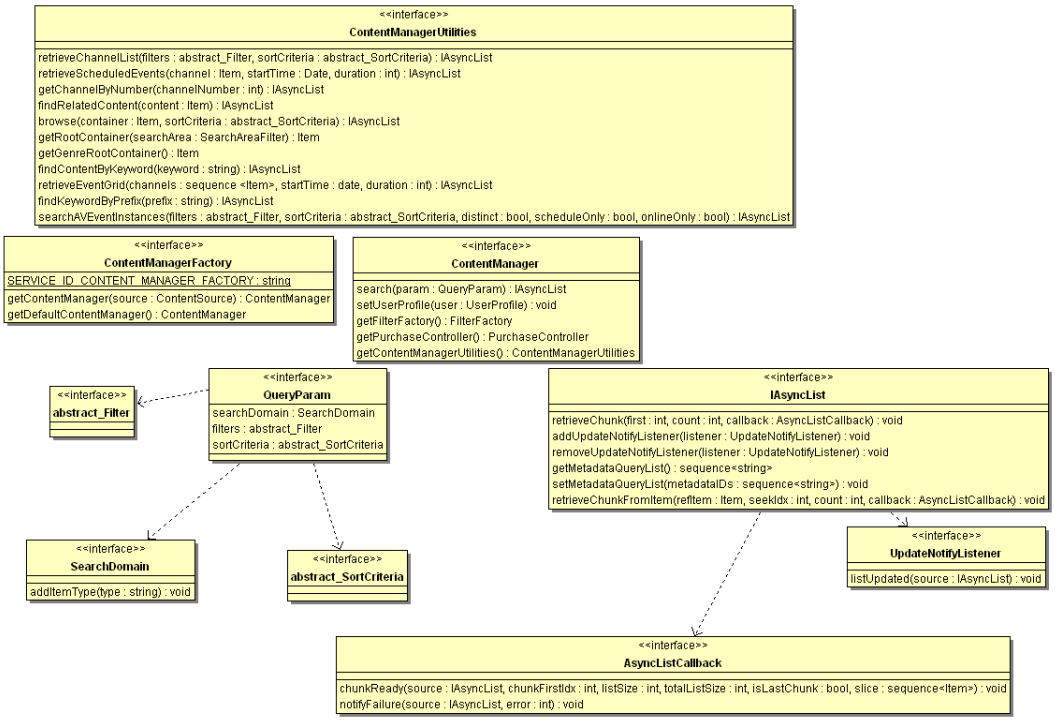


Figure 3 Entry point of the Content Cluster

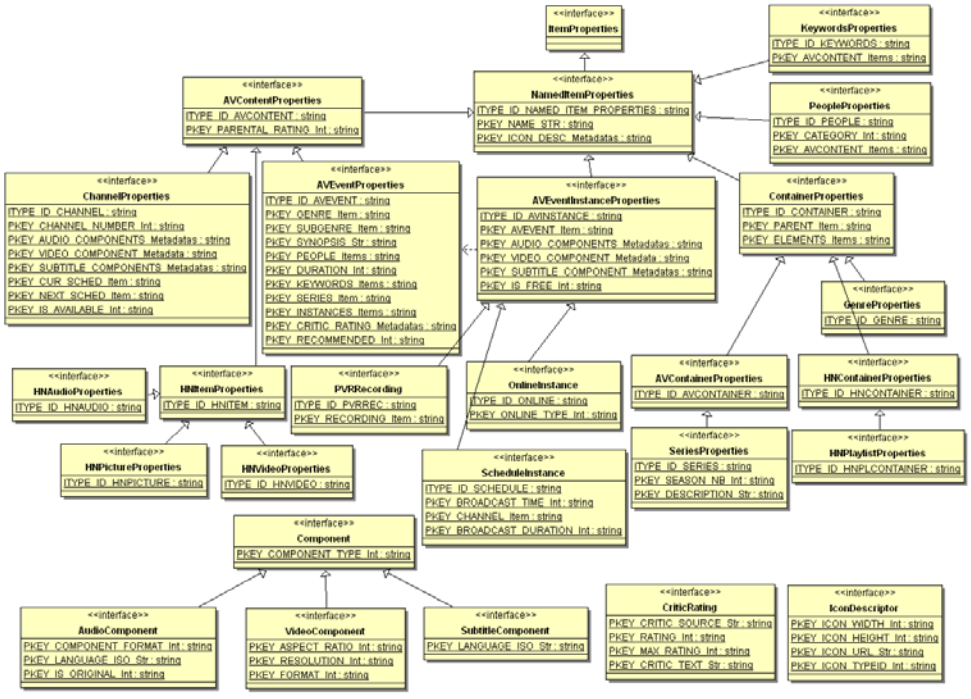


Figure 4 - Main Items

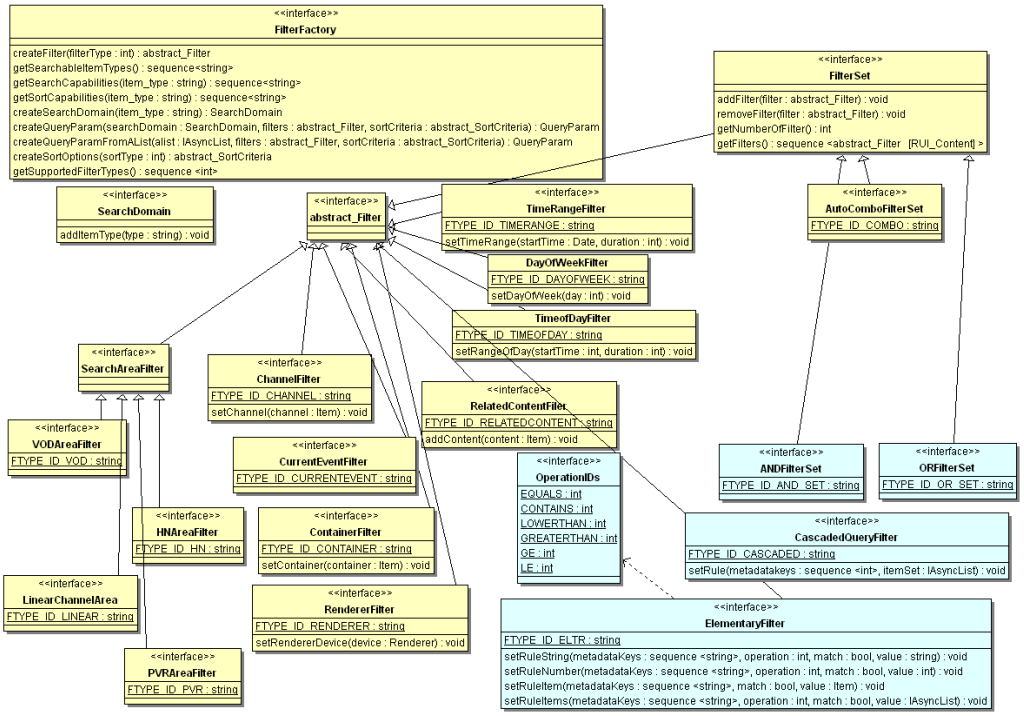


Figure 5 Filter Factory & filters

2. Device Cluster

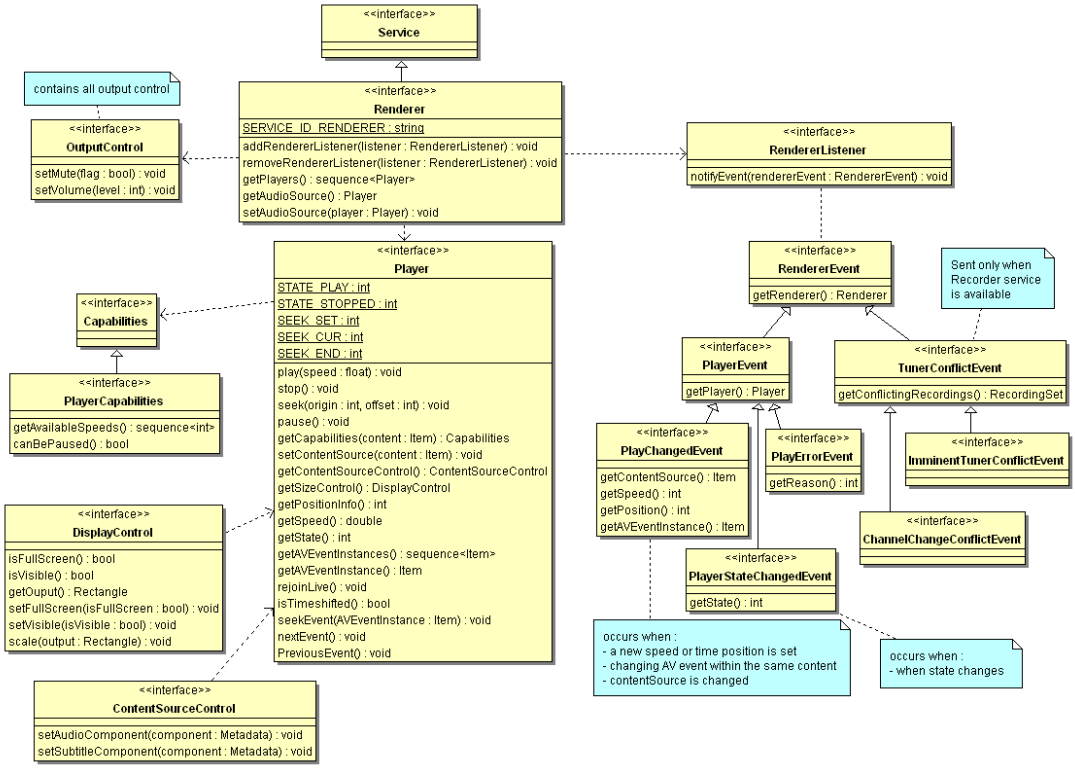


Figure 6 Device Cluster

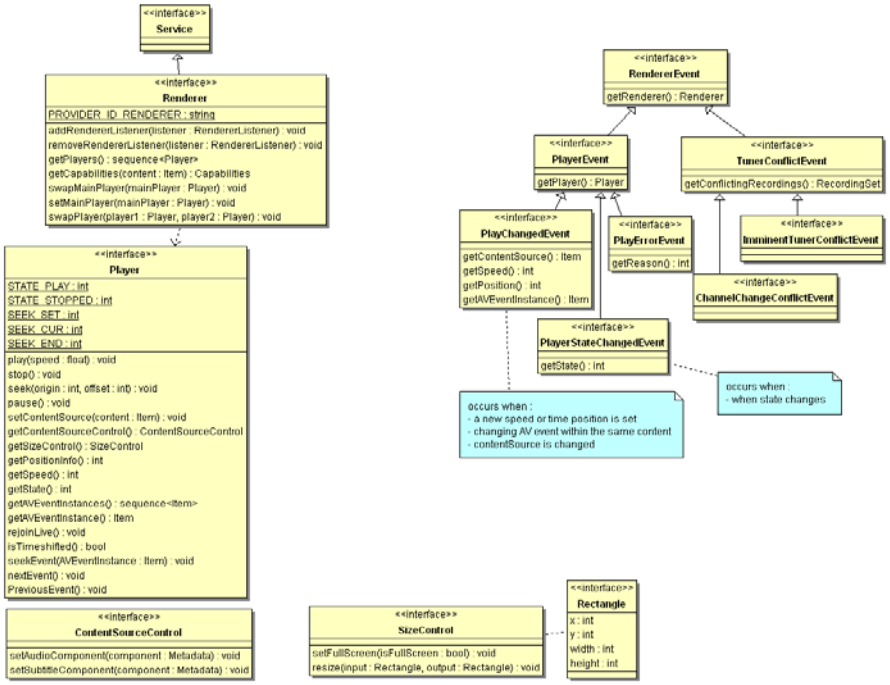


Figure 7 Events

3. Discovery Cluster

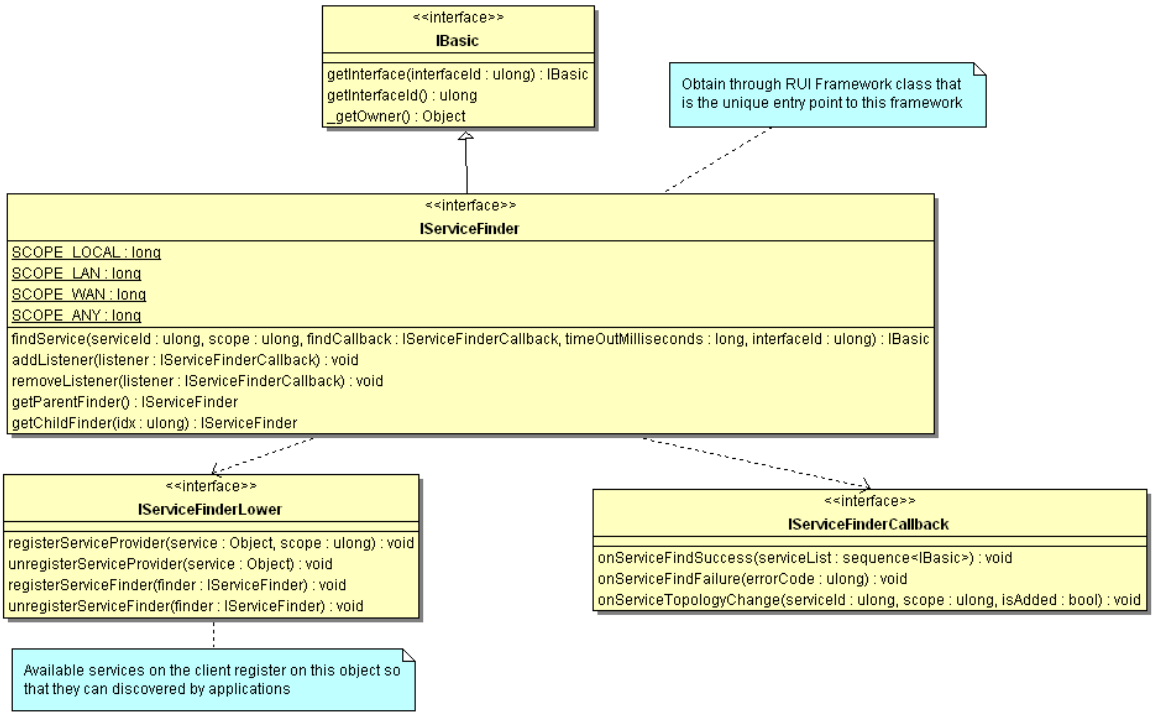


Figure 8 Discovery Cluster

Part two

This part is designed to convince a supervisor to hire me into a new project.

I choose to make the document as an email because is the most common way to communicate into NDS. For the background, I write this email to a fictive supervisor called John Doe, and it supposed to be sent after that I have spoken informally with Mr. Doe about his new Thor Project.

Vigie, Benjamin (prestataire)

From: Vigie, Benjamin (prestataire)
Sended: August, Wednesday 8 2012 10:48
To: Doe, Jhon;
Object: Postulation on the new Thor project

Hello Mr. Doe,

The RAPI project is drawing to a close. As you speak about the new Thor project, you know my interest for it. Indeed, it's typically the kind of stimulating and challenging project that I want to have for two main raisons.

The first one is the innovative and technical aspect of the project. You already work with me during my internship by give me some directions and responsibilities. So you know how much rigorous and efficient I am. Remember the Canal Plus demo page: I finish in time despite the very short development time you gave me. More of that, the Thor project is based on JavaScript, which is my favorite technology and the main I use during my internship. So Thor project fit perfectly on my skills.

The second thing is that I use to work with other Thor project's team members. We develop together some complicity and work methods, and if we can keep this complementarity I'm sure this will be benefic for a quick and valuable project development. There is a good atmosphere in our team and you know how much important is it in a project in term of motivation. And I very motivate and excited about this project.

I hope there is still a seat for me in this interesting and innovative new project. I'm available if you want some precisions about my work on other project at any time.

Vigié Benjamin
Ingénieur consultant Extia

Part three

This part is intended to a non-technical supervisor to try to convince him to give me the full responsibility of a project from the beginning to the end.

This time I choose to write a letter because it's more formal and appropriate for this kind of demand. Once again, the recipient and the project are fictive.

Benjamin Vigié
10, avenue de la fraternité
94200 Asnière
Phone: 06.78.80.68.37
Mail: benjamin.vigie@epitech.eu

Asnière, August25th, 2012

To Mr. Shin Chan
NDS France Vice President

Object:

Dear Mr. Chan,

As I notice you the last week, my internship is close to the end. I would officially ask you to be in charge of the LOLILOL project.

As you know, I worked during my internship on the RAPI project. This work included of course a development part but also a specification part. I had to work with architects to help them and give them some advices about the conception of the API. This was a great and rewarding experience; concluded by the sale of the product. I really like to be a reference during conception's meetings. It's an important part of a project, and it's typically the kind of work I want to develop my career. I'm organized and meticulous, and it is precious skills as team leader.

I am even more reassured that LOLILOL is based on JavaScript, a technology where I feel very comfortable. I demonstrate during my internship my use of this technology through the RAPI project. I was in charge almost alone of the project's development. I use this technology for a long time, and now I can give directions and advices for people who use it. I think a team leader has to have good development skills to be efficient. This position requires taking important decisions. I think today my technical background allow me to take more responsibilities on a project.

As I told you, I would have a new challenge and go further. I think LOLILOL project is the perfect project for me to have more responsibilities. So please considerer my offer as team leader on it.

Best regards,

Vigié Benjamin