



# Télécom ParisTech

## Internship Report

---

Data Engineer - Société Générale  
Project HUB – Pizza team BDDF

---



Ms. Phuong Pham  
M2 Data and Knowledge

Academic advisor: Ioana Manolescu  
SG advisor: Laurent Gendrier

March 2018 — August 2018  
Confidential Document

## **Privacy Notes**

This document contains information internal to Société Générale and is of confidential nature. Its usage by people other than the Université Paris-Saclay M2 Data and Knowledge 2018's Thesis Jury chairpersons are subjected to the authorization of the company.

Unauthorized people who possess either this documents or one of its copies are requested to destroy it and to inform its author or the company immediately. The usage of the information contained in this document or the documents attached to it by an unauthorized person is strictly forbidden.

## **Abstract**

This project is developed under the contexts at Société Générale with an aim to create a web application prototype for improving big data management quality. The web application will be able to take subscription from administrative users and handle automatic data transmission from data-lake to project's space. Whenever a subscription is made, the details of the subscription will first be written to the PostgreSQL database. The application will then query LDAP and Ranger for checking a user's right to access to files in the data-lake and to write them to his project's space. Next, it will consult the data catalogs in PostgreSQL database to show the user a list of data available for subscribing. When the user subscribes to some data, it will use Apache Spark to create hive tables using the data in the data-lake, and select suitable columns on these hives tables before writing them to project locations on HDFS. To schedule the transmission, it will use Oozie workflow. This project will change the way projects consume the raw data stored in the data-lake. It will also be the first step for industrializing the Data Distribution service of the Big Data Platform.

## **Acknowledgements**

This project cannot be completed with helps and support from many people. In particular, I am graceful to Dr. Ioana Manolescu – the academic advisor of this project - and Dr. Fabian Suchanek – the master’s program coordinator – for their support and their help which allowed the good progress of my training.

I also want to thank Mr. Laurent Gendrier – the supervisor of this project – for his patience to guide and support me throughout the project; your discussion, feedbacks and helping hand have been absolutely invaluable.

I also wish to express my special gratefulness for my colleagues, including Mr. Nicolas Cerutti, Mr. Mourad Karoui, Mr. Gaetan Mathias, Mr. Mohand Mamma, Mr. Florent Nallatamby, Mr. Wassim Arfi, Mr. Wandrille Domin, Mr. Julien Cordier, Ms. Massisa Dombolo and Mr. Jerome Ducourtioux for their constant availability to assist me in completing this project.

Lastly, I would like to thank the rest of the staff at the Big Data Computing Center that I do not have the opportunity to quote and other people from the Société Générale group that I met during my experience

## Table of Contents

Chapter 1 .....	7
Introduction.....	7
1.1    Problem statements and Approach .....	7
1.2    Goals.....	8
1.3    Functional requirements .....	8
1.3.1    Authentication to access to the platform .....	8
1.3.2    Search for a source data structure(s) .....	8
1.3.3    Subscription/Unsubscription action.....	8
1.3.4    Create internal table and write the table to the project directory .....	9
1.3.5    Schedule the process with oozie .....	9
1.4    Constraints .....	9
1.4.1    Files partition on the HDFS.....	9
1.4.2    The data catalog .....	9
Chapter 2 .....	11
Background and theory .....	11
2.1    HDFS (Hadoop Distributed File System) .....	11
2.2    Oozie.....	11
2.3    LDAP .....	11
2.4    Apache Ranger.....	11
2.5    CGI (Common Gateway Interface) .....	12
2.6    The CCB file format.....	12
2.7    Maven project .....	12
Chapter 3 .....	13
Design and Methodology .....	13
3.1    System designed.....	13
3.2    Methodology .....	17
Chapter 4 .....	27
Results .....	27
Chapter 5 .....	49
Conclusion .....	49
References.....	50

Appendix 1 – Manually compile HTTP Server .....	51
---	----

# Chapter 1

## Introduction

### 1.1 Problem statements and Approach

New data flows from different sources such as orders or deals are collected in the data-lake in a recurring manner. Subsets of such data flow have to be transmitted to a project's space for further exploitation either by an end-user or a third-part application.

Until now, each project needs access to the whole data source in order to be able to get a subset of data from this source. To transmit this subset of data, each project has to develop its own treatments: manually creating scripts generating hive tables for subsets selection, and scheduling the aforementioned process in a recurring manner. This has two consequences:

- Cost: Each project develops its own treatment for quite same purpose.
- Security: The project access to more data than those that are useful for its use case. As we have more and more data sources that are being accessed for more than one use case/project, exposing data using a service based approach is becoming more and more necessary.

On the other hand, we are evolving from building our solutions use case by use case to services that can be consumed by multiple use cases. The Big Data Platform will eventually deliver two main services:

- Data Exploration and Massive Computing Analysis
- Data distribution

For these reasons, this project will aim to change the way projects consume the raw data stored in the data-lake. In particular, it will standardize the way to deliver data for end-usage, and prevent the risk of having a lot of people accessing data that they are not supposed to access. What we develop here will be the first step for industrializing the Data Distribution service of the Big Data Platform. The data-lake layer will eventually become more and more unreachable without a manage service layer.

To address these needs, this project will develop a web application prototype that can automatically provide a project with subset(s) of data from a source collected at the lake in a right-ensuring manner. A subscription to files transmissions can only be made by an administrative user. Whenever a subscription is made, the details of the subscription will be written to the PostgreSQL database. The application will then query LDAP and Ranger for checking a user's right to access to the files in the data-lake and to write them to his project's space. Next, it will consult the data catalogs in PostgreSQL database to show the user a list of data available for subscribing. When the user subscribes to some data, it will use Apache Spark to create hive tables using the data in the data-lake, and select suitable columns on these Hives before writing them to project locations on HDFS. To schedule the transmission, it will use Oozie workflow.

The project will solve the aforementioned needs in two ways. First, we will no longer need to develop one files transmission application per project. All of the parameters needed for files transmission will be collected via the web application interface. Then, the decision on how these files

should be transmitted (i.e. which part of the Spark script to run, how the oozie workflow should be schedule) will be made using these parameters. Second, since files transmission will be automatized, we will be able to replace the jobs of many developers by a handful of administrative users. Thus, we will only need to allow a small number of administrative users to access to a particular data source, instead of having to allow all developers of those projects who need data from this source.

## **1.2 Goals**

The goal of this project is to implement a web application that allows administrative users to subscribe to or unsubscribe from schedulers. These schedulers will be responsible for automatically transferring subsets of data required by a project from sources in the data-lake to the project's space. The subsets of data required by a project will be defined by columns of interest, frequency and output file format. The transmission will be performed only when users have the right to read the content of the subscribed sources, as well as the right to write to the chosen project's spaces.

## **1.3 Functional requirements**

### **1.3.1 Users**

Administrative users

### **1.3.2 Authentication to access to the platform**

The LDAP service will need to verify an end user's identity when he accesses to the platform.

### **1.3.3 Search for a source data structure(s)**

Given the application code (code IRT), the platform should be able to display all the *file structures* and the description of their attributes that are collected by this source (this application). A file structure will represent all files from this source that has this data structure. For example, all files with naming convention PALGP.PHADP01M.ALHADCT.CATALE01.DJJMMAA.RECU, where DJJMMAA will be replaced by the current date, will contain 2 columns: "Alert category code" and "Description of the alert category"

The source data structure name and its sets of columns will come from the data management catalog.

### **1.3.4 Subscription/Unsubscription action**

An end user acting as an administrator should be able to select through an interface:

- Source data structure name
- A set of columns
- The frequency of updating
- The format of the extraction.
- The destination of the extraction (according to LDAP and rangers habilitation)

User's rights to make a subscription to a source data structure and to write to a project space should be verified by querying LDAP groups of the user and Ranger policies for a these groups. Once user's rights have been checked, the consequent actions will be executed under the right of an applicative account. The applicative account should have rights to access all the projects, and a set of application directories (updated on demand).



All the subscription should be stored in PostgreSQL database so that:

- An audit could be done
- A search for already existing subscriptions can be done
- A duplication of a subscription that already exists can be prevented. A duplication of subscription is defined as follow: One user can only subscribe to the same project space one source data structure with one subset of attributes.
- An updating of a subscription that already exists can be done
- An unsubscription can be done (by querying, stopping the coordinator id that currently takes charge of the scheduling and updating the PostgreSQL database)

### **1.3.5 Create internal table and write the table to the project directory**

An internal hive table should be created for each file in the subscribed file structure. Hive tables allow selecting the desired columns as well as performing some treatments to the files before sending it to the project's space (these treatments will not be done in this stage of the project). The input file formats supported are txt, csv, json, xml and ccb.

Selected columns should be written into the project directory under user's desired format. The supported output file formats are txt, csv, json and xml. The output files are guaranteed to be:

- Complete (no information is loss during the transmission) or user will be informed when the file is not.
- Transmitted only once:
  - If the user wants most recent files, only files that arrive in lake in the timeframe [today, (today - file\_arriving\_frequency)] will be transmitted.
  - If the user wants all files, only those that have not been transmitted, or those that have been transmitted but with a different subsets of columns will be (re)transmitted.

### **1.3.6 Schedule the process with oozie**

When new data arrives at the registered source, its subset will be automatically transferred to the user's project space, according to the subscription detailed made by that user.

## **1.4 Constraints**

### **1.4.1 Files partition on the HDFS**

In order for the automatic files transmission to work, files in data-lake have to be partitioned by /<entity\_name>/<application\_name>/<topic>/<date>.

### **1.4.2 The data catalog**

This project largely relies on the data catalog to work. Therefore, the catalog needs to have these following characteristics:

1. The catalog has to be written in utf8 format
2. Given an application's id, we need to be able to look up for the structureNames, structureLocations and structureFileFormat from the catalog.
3. Given a structureName and structureLocation, we need to be able to look up for the attributeNames, attributeDescriptions and attributeTypes from the catalog.
4. The naming convention of structureName is <prefix\_name>DDMMYYYY<suffix\_name>. For example, PALGP.PHADP01M.ALHADP\*.PARCLI01.DJJMMAA.RECU is a valid structureName

5. The naming convention of structureLocation is  
/lake/<entity\_name>/<application\_name>/<topic>/aaaammjj
6. Valid attributeTypes are: ArrayType, BinaryType, BooleanType, CalendarIntervalType, DateType, HiveStringType, MapType, NullType, NumericType, ObjectType, StringType, StructType, TimestampType
7. The attributeNames in the catalog have to be identical that of the input files

## **Chapter 2**

### **Background and theory**

#### **2.1 HDFS (Hadoop Distributed File System)**

HDFS is a distributed file system that runs on commodity hardwares – hardwares that are relatively inexpensive and can easily be replaced in case of failure. An important characteristic of HDFS is that it consists of many servers machine, each of which stores (or replicates) a part of the file system's data. Each machine is also responsible for executing computation on the data that it hosts in parallel with its peers. Another characteristic of the HDFS is that it provides high throughput access to application data and therefore is suitable for applications that have large data sets. It is also portable from one platform to another, which makes the adoption of HDFS becomes much easier.

A HDFS cluster consists of a single namenode and several datanodes. Each file in the HDFS is split into smaller blocks. Namenode has two properties. First, it manages the file system namespace, which is a catalog of data blocks' directories across the file system. Second, it monitors access to files requested by a client application. Datanodes physically store the data blocks and their replications, as well as serve the read and write requests from the client application.

#### **2.2 Oozie**

Oozie is a workflow scheduler system for Hadoop. A workflow contains control flow nodes and a collection of action nodes arranged in DAG (Direct Acyclic Graph). Nodes in the workflow are control dependency, which means the second action only executes when the first action has completed. Control flow nodes either mark the beginning and the end of the workflow ("start", "end" and "fail" nodes) or provide a mechanism to control the workflow execution path ("decision", "fork" and "join" nodes). Action nodes mark an execution or a computation task to be done. A workflow executed on a recurring basis can be specified as a coordinator application in Oozie.

#### **2.3 LDAP**

LDAP (Lightweight Directory Access Protocol) is an extensible open network protocol standard that provides access to distributed directories. These directories contain objects, generally those related to users, groups, services, etc. LDAP provides a method to authenticate clients to the directory server (method bind()), and a method to retrieve partial or complete copies of entries matching a given set of criteria (method search()). The bind method will be useful for us to authenticate user access to the application, whereas the search method will be useful for us to query user's groups and thus allow us to check whether the user has the right to access to certain locations on the HDFS.

#### **2.4 Apache Ranger**

Apache Ranger is a framework that allows security administrators to define and manage security policies consistently across Hadoop components. Security policies define users' access right to one of the following resources: files, folders, databases, tables, or column on the HDFS. Administrative users can interact with Ranger using REST API [1]. REST API contains a set of functions that allow administrators to perform requests and receive responses via HTTP protocol such as GET and POST. For example, in order to retrieve all policies on the HDFS, we can call "GET policy" protocol on one of

the HDFS servers: <https://dhadlx102.haas.socgen:6182/service/public/api/policy>. The returned result will be in JSON format.

## 2.5 CGI (Common Gateway Interface)

CGI is a protocol that allows web servers to execute console applications such as python scripts or c executable files. Such applications are known as CGI scripts.

A web server allows its owner to configure the URL that triggers the execution of the CGI scripts. This is often in the form of `http://<server_address>/cgi-bin/<script_name>?<script_input_parameters>`, which points to the CGI scripts located at `cgi-bin`. When a browser requests such URL, the HTTP server will execute the specified script taking into account the input parameters, and then pass the output of the script to the Web browser.

## 2.6 The CCB file format

The CCB file format is an SG internal file format. Each CCB data file's schema is defined by another file called "clause copy". Consider the following data and clause copy.

data.txt	clause_copy.txt
A0877300030099100120170523152104A	15 WW00-CDAPIN-B PIC X(5).
131785	15 WW00-IDELSS-B PIC 9(10).
	15 WW00-IDCADI-B PIC X(3).
	15 WW00-DASAO9-B PIC X(8).
	15 WW00-HESAO2-B PIC 9(6).
	15 WW00-IDAGNT-B PIC X(7).

This means that the first 5 characters of the data are of type X (String) and belong to column WW00-CDAPIN-B, the next 10 characters are of type 9 (Integer) and belongs to column WW00-IDELSS-B, and so on.

## 2.7 Maven project

Maven is a tool for managing project's build. It defines how object oriented scripts such as .java and .scala get compiled into .class and packaged into .jar files. Maven downloads all the libraries that are required to be used by the project automatically. It also has many plug-in for .java and .scala scripts that we can install by simply adding them to pom.xml. Maven also "knows" how to "compile", "test", "package", and "clean" by itself. That is, we do not have to write the scripts for these tasks. Instead, we can just put the files at the places in which Maven expects them and it should work. For the above reasons, we will develop this application as a maven project.

## 2.8 Other tools/technologies

In accordance to the existing context at SG, we will use PostgreSQL database for storing the data catalog as well as the subscriptions made by users; and Spark for manipulating files in the HDFS.

## Chapter 3

### Design and Methodology

#### 3.1 System designed

This section will give a detailed explanation of the whole system, including the data on which this application depends and the overall architecture.

##### 3.1.1 Data on PostgreSQL Database

###### 3.1.1.1 Data catalog

The data catalog provides information about the file structures that each application has, including their application lake (resp\_application\_lake), data type (data\_type), location on the HDFS (directory), naming conventions (file\_name), formats (file\_format), delivering frequency to the HDFS (frequency) and clause copy if any (clause\_copy). The data catalog is registered in PostgreSQL database as hub\_data\_catalog table. This table is created from hub\_data\_catalog.csv file which is located at /home/A469255. This file is being updated every so often by the data management team.

An extraction of the data catalog is shown below. Recalls that file structure will represent all files from this source that has this data structure. The file structure PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU refers to all files whose name begins with “PALGP.PHADP01M.ALHADAL.ALELUE01.”, ends with “.RECU” and modify DJJMMAA to be the current day.

```
habibgd=>
habibgd=>
habibgd=> select * from hub_data_catalog where file_name='PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU';
 resp_application_lake | data_type | directory | file_name | cd_irt
| frequency | compression | file_format | clause_copy
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
 bddf | Donnees Bancaires | /lake/bddf/a5103_alg/alertes_lues/aaaammjj | PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU | a5103
| Quotidienne | Non | XML |
(1 row)

habibgd=> █
```

### 3.1.1.2 Data attributes catalog

Data attributes catalog provides more detailed information about a file structure. In particular, it provides information on the column names (attribute\_name) and their description (attribute\_description) in addition to what already is available in the data catalog. The data attributes catalog is registered in PostgreSQL database as hub\_data\_catalog\_attributes table. This table is created from hub\_data\_catalog\_attributes.csv file which is located at /home/A469255. This file is being updated every so often by the data management team. An extraction of the data attributes catalog is shown below.

```
hakilbgd=> select * from hub_data_catalog_attributes where file_name='PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU';
```

directory	file_name	attribute_name	resp_application_lake
/lake/bddf/a5103_alg/alertes_lues/aaaammjj	PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU	Identifiant de l'alerte generee	
/lake/bddf/a5103_alg/alertes_lues/aaaammjj	PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU	Code canal de lecture de l'alerte a partir duquel l'alerte a ete lue, pouvant prendre les valeurs suivantes: WEB, APPLI, WEBMOB	
/lake/bddf/a5103_alg/alertes_lues/aaaammjj	PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU	Date et heure de lecture de l'alerte sous le format : JJ/MM/AAAA HH:MM:SS	

(3 rows)

### 3.1.1.3 Subscription database

The subscription database stores the detail of each subscription, including the user who made the subscription (user\_id), the project space (project), the registered structure (registered\_structure), the registered columns on this structure (registered\_attributes), the arriving frequency of the files of this structure (frequency), the validity of this subscription (valid), the id of the coordination who is in charge of the scheduling (coordid), the subscribe date (registered\_date) and the unsubscribe date (cancel\_date). The data attributes catalog is registered to PostgreSQL database as hub\_registration table. An extraction of the subscription database is shown below.

```
hakilbgd=> select * from hub_registration;
```

userid	project	registered_structure	registered_attributes	frequency	valid
coordid	registration_date	cancel_date			
A469255	/project/bddf/hub/storage/test	PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU	id_alerte,canal_lecture	Mensuelle	yes
0003677-180716145158909-oozie-oozi-C	2018-07-20T14:00+0200	999			

(1 row)

### **3.1.2 Architecture overviews**

An overview of the project architecture is shown in figure 1.

#### **3.1.2.1 SG's HDFS files collection and organization**

##### **3.1.2.1.1 TOM service**

TOM is internal software at Société Générale, which is in fact an IBM product named IBM Sterling Connect Express. This product is a file transfer monitor, which means in addition to files transferring, it also offers transfer supervision functions. Connect Express performs files transmissions using the PeSIT6 protocol. PeSIT (Exchange Protocol for Interbank Compensation Scheme) is a protocol used for transferring files between two computer systems connected by a telecommunication link. This protocol allows computer systems to exchange files with partners who have different transferring monitors, providing that they use the PeSIT protocol.

##### **3.1.2.1.2 Data collection**

Different types of data: structured, semi-structured and non-structured data that comes from either internal or external sources at the bank are transmitted either synchronously or as mini-batches to data-lake via TOM service. The data from the same source may arrive at the data-lake in a repeating manner depending on the function of that source. For example, transactions made by an agency during a day will arrive at the data-lake daily.

In Big Data environment, the data collected by TOM is not directly integrated into the data-lake. Rather, they will first be buffered in Virtual Machines called edge servers before being sent to the data-lake in order to ensure the security of the Hadoop cluster with respect to the external environment.

##### **3.1.2.1.3 Data organization**

The SG's HDFS is hosted by different servers, namely dhadlx110, dhadlx112, dhadlx113. Files stored in the HDFS are partitioned by category. In this project, we only pay attention to the lake and the project category.

Files in lake are first partitioned by entity names then by application ID. They will be further partitioned by topics and the date of collection. For example, files at location `/lake/bddf/a0877_urta/ref_transactions/20180626` means that they belong to the BDDF entity, they come from the URTA application (source), they are transaction references and they were collected on 26/06/2018.

Similarly, files in project are first partitioned by entity names then by project name. They will also be further partitioned by topics and the date of collection.

##### **3.1.2.2 The architecture of the application**

Users will interact with the hub application interface to specify his demand. This application runs on an internal apache server `http://dhadlx112.dns21.socgen:8090`. This is also one of the servers that host the HDFS. When a demand is made, the apache server will trigger the appropriate CGI script which will serve the demand.

## Projet Hub – Architecture

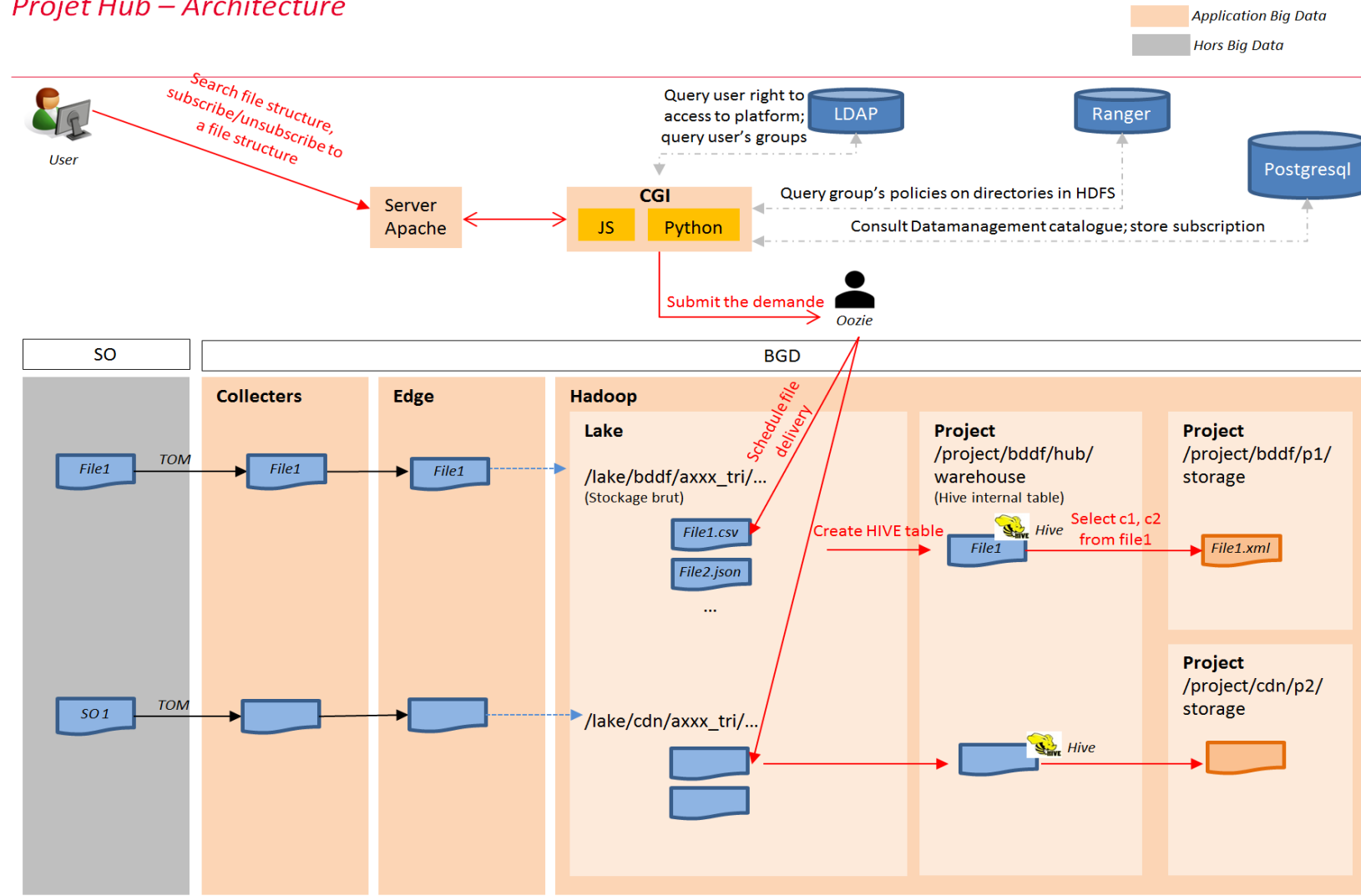


Figure 1 - Architecture



A JavaScript is responsible for parsing user inputs from html forms and sending them to the appropriate python program. The python program can first use cgi library to load and read these input parameters then perform further operations. These operations include: authenticate the user, search for file structures and their attributes, and serves the subscribe/unsubscribe request. Subscribe/unsubscribe request will be served by an oozie coordinator application, which will execute a jar that contains the spark jobs that are responsible for file treating and transferring in a recurring manner. The oozie coordinator application will be launched when the python script calls a bash script.

Spark 2.11 is used to develop this application.

### 3.2 Methodology

In this session, we will discuss the application of the technologies described in chapter 2 on the development process

#### 3.2.1 Environmental setup

The first step of our development is to install an HTTP server on one of the machines in the HDFS. I chose to install it on machine dhadlx112.dns21.socgen for the following reason. This machine hosts the PostgreSQL database and thus has the libraries that support the interaction between PostgreSQL and python. Due to the security complication, I had installed the HTTP server manually using the following sources:

- apr-1.6.2.tar.gz
- apr-util-1.6.0.tar.gz
- expat-2.2.1.tar.gz
- httpd-2.4.26.tar.gz
- pcre-8.41.tar.gz

The steps of the installation can be found in appendix 1.

#### 3.2.2 Parsing user inputs from html forms and send them to CGI scripts using JavaScript

The **login.html** is the login page of the application. It consists of 1 function:

- show

**show.** show function is activated when user clicks “Login” button. It does the following:

1. Parse username and password
2. Check if one of the required fields is left blank, if yes ask the user to enter them
3. Call the authentication.py script which will authenticate the user
4. Load hub.html if the user is authenticated. Deny the user access to hub.html otherwise

The **hub.html** is the page that allows user to search for file structures of an application and subscribe/unsubscribe to these file structures. It consists of 4 main functions:

- xmlhttpPost
- popup
- postValues
- u\_postValues

**xmlhttpPost.** This function is triggered when the user presses the “Search” button. It does the following:

1. Parse username and applicationCode
2. Call the list\_structure.py which will check whether the user has the right to see the contents of that application, and if yes, list all the structures that this application has
3. Return the results from executing list\_structure.py to the user

**popup.** This function is activated when user presses the “Subscribe” or the “Modify” button in the file structure list HTML table. “popup” does the following:

1. Send the structureName and structureLocation to list\_attributes.py
2. Display the attributes returned by list\_attributes.py for each structure to the user

**postValues.** This function is activated when user presses the button “Subscribe” in the popup. postValues does the following:

1. Get the structureName, selectedAttributes, deliveringFrequency, outputFileFormat, projectDirectory and transferType (whether to transfer only most recent file or to transfer everything) from the user
2. Check if one of the required fields is left blank, if yes, ask the user to enter a value
3. Check if the user input frequency (deliveringFrequency) is larger than the actual file delivering frequency, if yes ask the user to modify the deliveringFrequency
4. Call the registration.py which will write/update the user subscription to the registration database (PostgreSQL db) and trigger the oozie workflow to schedule the file delivery

**u\_postValues.** This function is activated when a user presses the button “Unsubscribe” in the popup. u\_postValues does the following:

1. Get the username, structureName to which the user wants to unsubscribe, and the projectDirectory to which the user wants to stop the structures from being delivered
2. Check if one of the required fields is left blank, if yes, ask the user to enter a value
3. Call unsubscribe.py, which will update the user subscription to the registration database (PostgreSQL db) and stop the oozie coordinator who is currently in charge of this file structure delivery querying PostgreSQL database

### 3.2.3 The CGI scripts

#### 3.2.3.1 Authenticate the user

**authentication.py** is responsible for authenticating the user. It first gets the username and password from the login.html form using cgi library. Then, in order to check for user identity, the script will try to connect to the “User” Organizational Unit of LDAP using this username and password. If the connection is successfully established, it means the username and password provided by the user match that of the LDAP database. In this case, the user will be authenticated.

### 3.2.3.2 Search for file structures and their attributes

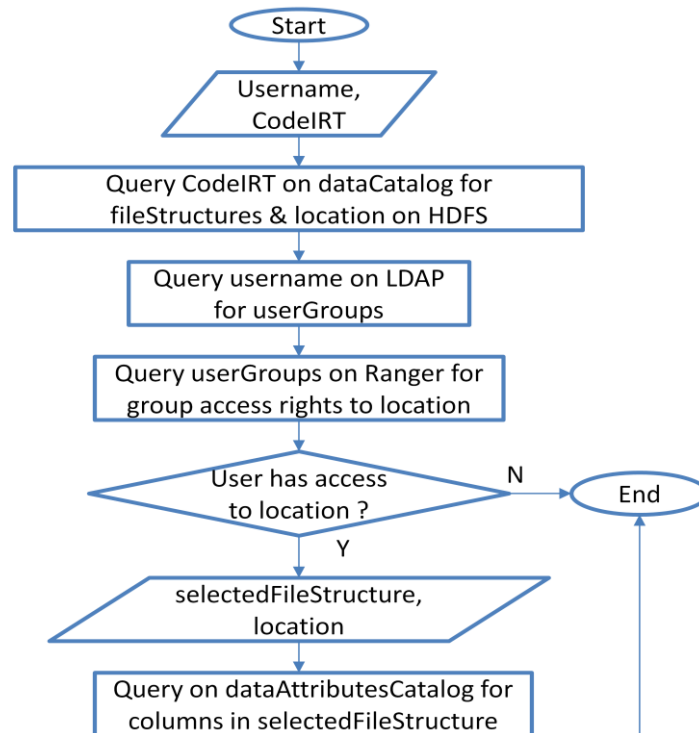


Figure 2 - List structures and their attributes

The overall process of listing structures and their attributes is summarized in figure 2.

**list\_structures.py** is responsible for listing file structures giving the application IRT code. It does the following:

1. Get the username and applicationCode from hub.html using cgi library
2. Query the applicationCode on data catalog in order to get the structureNames and structureLocations in the HDFS.
3. Connect to the LDAP server as an administrative user.
4. Query LDAP (by using ldap library) to get all the groups to which the user belongs.
5. Query ranger policies:
  - Filter only policies for the HDFS locations that are resulted in step 2, and for the groups that are resulted in step 4
  - Check if one of the groups has the rwx (read, execute, write) right on these HDFS locations
6. If the user has the right to see the contents of this application:
  - Connect to PostgreSQL database
  - Create or update the hub\_data\_catalog, hub\_data\_catalog\_attributes in this database
  - Query structureNames, structureLocations, deliveringFrequency, inputFileFormat, clauseCopy using the given applicationCode on the hub\_data\_catalog
  - Try to query structureNames, structureLocations and valid='yes' on hub\_registration to see if this structure has already been subscribed to. If it has not been subscribed to, the number of results should be 0.

- Print structureNames, structureLocations, deliveringFrequency, inputFileFormat, clauseCopy as a HTML table:
  - If the structure has already been subscribed to, make “Modify” and “Unsubscribe” button appear for this line
  - If the structure has not been subscribed to, make “Subscribe” button appear for this line

**list\_attributes.py** is responsible for listing a file structure’s attributes. It does the following:

1. Connects to postgresql database
2. Get file structureLocations and structureNames from hub.html using cgi
3. Query attributes from the hub\_data\_catalog\_attributes using structureLocations and structureNames
4. Print the query result as a check list

### 3.2.3.3 Serve the subscribe/Unsubscribe using oozie workflow and spark

Figure 3 summarizes the “subscribe/unsubscribe phase”.

#### 3.2.3.3.1 Subscribe

The subscribe phase consists of 3 steps: trigger the oozie workflow, oozie workflow deployment and the actual create/write Hive table step.

##### 3.2.3.3.1.1 Trigger the oozie workflow application

Once the user has chosen the structures and the attributes that he wants to subscribe to, the apache server will submit the parameters of the demand to Oozie and trigger Oozie to start scheduling the recurring file transmission process. This is done using the CGI script **registrationDB.py** as follow:

1. Get username, projectDirectory, structureName, structureLocation, structureAttributes, deliveringFrequency, inputFileFormat, outputFileFormat, clauseCopy, transferType from hub.html using python’s cgi library
2. Connect to PostgreSQL database
3. Create table hub\_registration if it doesn’t exist
4. Query table hub\_registration for coordinatorId that are still valid (valid=yes ) and have the same structureName and structureLocation as in step 1
5. Kill the coordinatorId resulted from step 4
6. Set valid=no and cancel\_date=today to all the lines resulted from step 4
7. Update file workflow.xml and coordinator.xml with the inputs from step 1 and with the current date. workflow.xml and coordinator.xml define the workflow to be scheduled by oozie.
8. Deploy the oozie workflow app using the deploy script (deploy-bin.sh). This step will result in a currentCoordinatorId
9. Update hub\_registration with new registration data (which are the inputs from step 1) and currentCoordinatorId
10. If step 9 fails:
  - Kill the currentCoordinatorId resulted from step 8.
  - Inform user that the registration with inputs from step 1 are failed to be subscribed to.

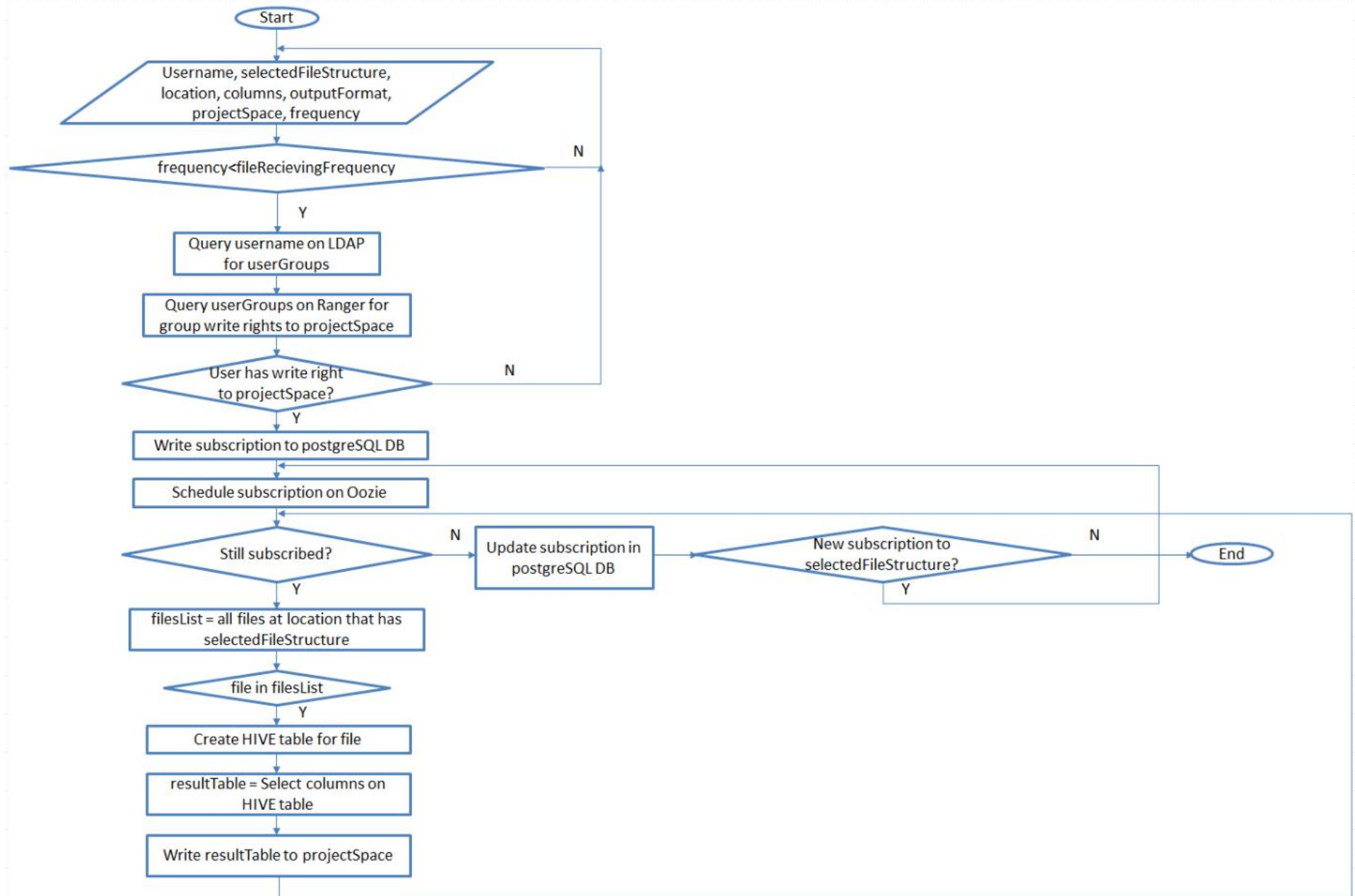


Figure 3 - Subscribe/Unsubscribe

### 3.2.3.3.1.2 Defining the oozie coordinator application

The oozie coordinator application is defined using 2 main files:

- coordinator.xml
- workflow.xml

**coordinator.xml.** This file defines the coordinator engine system for running the jobs in our workflow. Particularly, it defines the coordinator application, which includes the following: the coordinator app's name, recurring frequency, starting date (the current date), ending date, time zone, the path of the workflow app (**app-path**) and its configurations (**configuration**) such as the date to launch the app

**workflow.xml.** This file defines the workflow application, including its name, properties; the credentials to execute the app's jobs, the nodes in the workflow and their arguments.

Below is an extraction of the oozie workflow in our project. The action node is "Insert\_Table\_To\_Project\_Space". The control nodes are "start", "end" and "kill" where "end". The workflow goes to "end" if the action node is successfully executed and to "kill" otherwise.

```
<start to="Insert_Table_To_Project_Space"/>
<action name="Insert_Table_To_Project_Space" cred="hive_cred">
  <shell xmlns="uri:oozie:shell-action:0.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="uri:oozie:shell-action:0.2 ">
    <job-tracker>yarn-cluster</job-tracker>
    <name-node>hdfs://dhadcluster02</name-node>
    <exec>launch_spark.sh</exec>
    <argument>com.socgen.hub.spark.InsertTableToProjectSpace</argument>
    <argument>{INPUTDATA}</argument>
    <argument>{CLAUSECOPY}</argument>
    <argument>{INPUTFILEOPTION}</argument>
    <argument>{OUTPUTFILEOPTION}</argument>
    <argument>{OUTPUTLOCATION}</argument>
    <argument>{ATTRIBUTES}</argument>
    <argument>{DIRECTORY}</argument>
    <env-var>HADOOP_USER_NAME=hubpdadm</env-var>
    <file>/project/bddf/hub/apps/bash/launch_spark.sh#launch_spark.sh</
file>
    <file>/project/bddf/hub/apps/bash/hub-jar-with-
dependencies.jar#hub-jar-with-dependencies.jar</file>
    <capture-output/>
  </shell>
  <ok to="end"/>
  <error to="kill"/>
</action>
<kill name="kill">
  <message>Job killed, error
message[${wf:errorMessage(wf:lastErrorNode())}]</message>
</kill>
<end name="end"/>
```

### 3.2.3.3.1.3 Create/write Hive table

The oozie workflow will run the spark jar with the user's input (the input from step 1 of the registrationDB.py) on a recurring basis. The spark jar does two main things: create hive internal table

from input files, and write those tables into the user project directory. This jar is created with the scripts listed below. These scripts will be “packaged” as a jar file using maven.

- CobolParser.java
- CobolVariableSentence.java
- GenericCobolSentence.java
- HiveTableGenerator.java
- InsertTableToProjectSpace.scala
- CreateTable.scala
- Writing.scala
- compile\_jar.xml
- launchSpark.sh

The 4 java scripts are responsible for parsing the ccb data file and clause copy in order to get its schema and the data that is associated with this schema.

**CobolParser.java.** Class CobolParser consists of 1 method **findAllDeclarations**, which returns a linked list of CobolVariableSentence. Each CobolVariableSentence is a line in the ccb clause copy (a column’s name).

**CobolVariableSentence.java.**Class CobolVariableSentence and has the following important additional methods:

- getVariableName
- getVariableTypeSizeIfAny

**getVariableName** gets the variable name in the CobolVariableSentence. For example, if the sentence is “15 WW00-CDAPIN-B PIC X(5).”, getVariableName will return “WW00-CDAPIN-B” as variable name.

**getVariableTypeSizeIfAny** gets the variable size in the CobolVariableSentence. For example, if the sentence is “15 WW00-CDAPIN-B PIC X(5).”, getVariableName will return 5 as variable size.

**HiveTableGenerator.java.** Class HiveTableGenerator consists of 1 method makeSource, which does the following:

1. Get table schema from ccb clause copy

Initialize tableSchema

For each CobolVariableSentence in the first line of ccb clause copy:

Get the CobolVariableSentence name (column’s name) and add it to tableSchema

2. Get data in each rows from ccb data file and ccb clause copy

Initialize rows

Open and read the ccb data file

For each line in the ccb data file (lineDataFile):

Initialize startIndex = 0

Initialize row

For each CobolVariableSentence in the ccb clause copy:

Get the CobolVariableSentence size (the number of characters in ccb data file that belongs to this column)

```

        Add lineDataFile.substring(startIndex,size+startIndex) to row
        startIndex = size+startIndex
    Add row to rows

```

The 3 scala scripts are responsible for creating Hive tables from input files and writing these tables into the projectDirectory.

**InsertTableToProjectSpace.scala.** This is the main object of the jar file. It does the following:

1. Initialize SparkSession and get hadoopFileSystem
2. Get projectDirectory, structureName, structureLocation, structureAttributes, inputFileFormat, outputFileFormat, clauseCopy, frequency and transferType from oozie workflow application.
3. Call method createInternalTable() in object CreateTable to create hive tables for all files at structureLocation in the HDFS that has structureName. For each of these files, returns also the number of lines that they originally contain.
4. Get a list of all files at structureLocation in the HDFS that has structureName
5. For each of these files:
  - Check whether the entered structureAttribute is a nested attribute. If yes, we have to add the parent attribute to the name of the attribute. For example, if the column canaux\_pull has the following data {"code\_canal\_pull":["WEB","APPLI","WEBMOB "]} and the entered structureAttribute is code\_canal\_pull, we will have to fix the attribute name to canaux\_pull. code\_canal\_pull. This is done using method flattenSchema:

```

def flattenSchema(parent: String, schema: StructType):
Seq[String] = {
    schema.fields.flatMap {
        case StructField(name, inner: StructType, _, _)
=> {if (parent != "") {Seq(parent+"."+name) ++
flattenSchema(name,inner)}else{Seq(name) ++
flattenSchema(name,inner)}}
        case StructField(name, ArrayType(inner:
StructType, _), _, _) => {if (parent != "") {Seq(parent+"."+name)
++ flattenSchema(name,inner)}else{Seq(name) ++
flattenSchema(name,inner)}}
        case StructField(name, _, _, _) => {if (parent
!= "") {Seq(parent+"."+name) }else{Seq(name) }}
    }
}

```

flattenSchema recursively checks if an attribute (type structField or arrayType) is nested inside another attribute (type structFile). If yes, it re-executes method flattenSchema on the newly discovered attribute. At the same time, each time an attribute is discovered, its parent attribute will be concatenated with it. A parent attribute and a child attribute are separated by a dot '.'. The method returns all concatenated attributes.

- Select structureAttribute from the table created by createInternalTable() to create new Hive table. Compare the number of lines in the newly created Hive table with the number of lines returned from step 3.
- Call method writeToHDFS() in object Writing to write these tables to projectDirectory



**CreateTable.scala** is responsible for creating hive internal tables for the input files. It does the following:

1. Get structureName and structureLocation
2. Get a list of all files at structureLocation in the HDFS that has structureName
3. For each of these files:
  - Create a dataframe from the input file:
    - For xml files: By using “SQLContext.read.format(“com.databricks.spark.xml”)” to read the input file
    - For csv/text files:
      - Read input file and create inputRdd
      - Take the first row of the inputRdd and create the table schema
      - Take the next row of the inputRdd and create rowRdd
      - Create dataframe using the schema and rowRdd
    - For json files: By using “SQLContext.read.json()” to read the input file
    - For ccb files:
      - Create the table schema from clause copy and rowRdd from data file using makeSource method in class HiveTableGenerator()
      - Create dataframe using the schema and rowRdd
  - Create a hive table and insert data to this table using the dataframe created from the previous step.

**Writing.scala**. This script contains two methods:

- writeToHDFS
- write

**writeToHDFS** does the following:

1. If the user only wants to transmit most recent file:
  - Compute previousTransferDate = today – deliveringFrequency
  - Compare if the file delivering date is between today and previousTransferDate. If yes, call write method on this file.
2. If the user only wants to transmit all files:
  - Check if the files to be transmitted has already existed in projectDirectory:
    - If the name and size of the file to be transmitted is similar to one of the files at projectDirectory, we can say that the file to be transmitted has already existed in projectDirectory.
  - If not, call write method on this file.

**write** does the following:

If the output file format is csv/txt: By using “write.format(“com.databricks.spark.csv”)” to write the hive tables created by CreateTable.scala to projectDirectory

If the output file format is xml: By using “write.format(“com.databricks.spark.xml”)” to write the hive tables created by CreateTable.scala to projectDirectory

If the output format is json: By using “write.format(“json”)” to write the hive tables created by CreateTable.scala to projectDirectory

Method write name the new file as follow:

<original\_file\_name><file\_size><complete/incomplete>

**launchSpark.sh** configures spark-submit environment and executes the spark jar

#### **3.2.3.3.1.4 Deploy the oozie workflow**

Maven will also “package” the whole project as a zip file for deployment. In this zip file, the project’s tree will be respected. The project deployment will be done using deploy-bin.sh.

**deploy-bin.sh** sets up the deployment environment and then triggers the oozie coordinator application to execute. It does the following:

1. Get arguments (execution environment: development/production, execution mode: deploy/undeploy)
2. Define the appropriate (logPath, oozieUrl, kerberosTicket) for the chosen execution environment
3. Initialize Kerberos ticket for project HUB
4. In case of deploy:
  - Upload all resource scripts (generated from the zip file) to the HDFS. Log the process to a reportFile at the same time
  - Trigger the oozie coordinator application using the resource files on the HDFS. Log the process to a reportFile at the same time
5. In case of undeploy
  - Delete all resource scripts from the HDFS

#### **3.2.3.3.2 Unsubscribe**

**unsubscribe.py** is responsible for serving the unsubscribe request. It does the following:

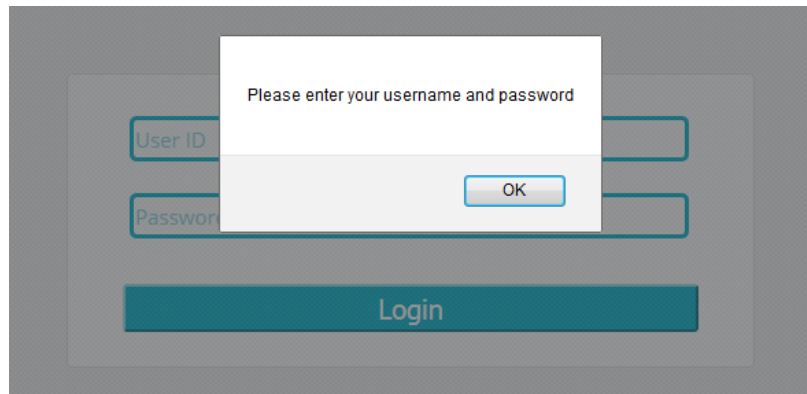
1. Get username, projectDirectory and structureName from hub.html using python’s cgi library
2. Connect to PostgreSQL database
3. Query coordinatorId from hub\_registration table using the input username, projectDirectory and structureName
4. Kill the coordinatorId resulted from step 3
5. Update the hub\_registration table (set cancel\_date to today and the validity of this subscription to ‘no’)
6. If step 5 fails: Inform user that the coordinatorId has been killed but hub\_registration has failed to be updated, and tell him to report this error to an admin.

## Chapter 4

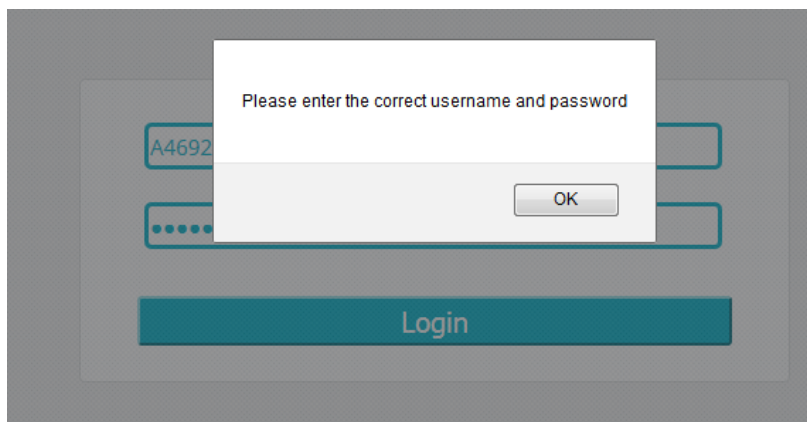
### Results

The project has met all the functional requirements specified in session 1.3 , as demonstrated in the example below.

When enters the platform, the user is required to enter the correct username and password. If the username and password are blank, the user will be informed.



If the username and password are left blank, the user will also be informed.



Once a user is authorized to enter the platform, he can start to search for file structures:

# HUB

[A469255](#)

Directory	File Name	Frequency	File Format	
/lake/bddf/a5103_alg/types_alertes/aaaammjj	PALGP.PHADP01M.ALHADTY.TYPALE01.DJJMMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/cat_alertes/aaaammjj	PALGP.PHADP01M.ALHADCT.CATALE01.DJJMMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/param_client/aaaammjj	PALGP.PHADP01M.ALHADP*.PARCLI01.DJJMMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/alertes/aaaammjj	PALGP.PHADP01M.ALHADA*.ALEGEN01.DJJMMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/param_client_supp/aaaammjj	PALGP.PHADP01M.ALHADPS.PARSUP01.DJJMMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/alertes_lues/aaaammjj	PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>

If the user doesn't have the right to see the content of the application he requested, he will be notified:

# HUB

[A469255](#)

You do not have the right to see the contents of this application. Please contact your administrator.

When a user wants to subscribe to a file structure, he will be asked to select the file's attributes and to enter the project directory, file delivering frequency and output file format.

HUB

A469255

Directory

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a5103\_alg/alertes\_lues/aaaammjj

x

Project Directory

Frequency

Output Format

☐ Only transfer most recent files

Subscribe

PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU

☐ Identifiant de l'alerte generee

☐ Code canal de lecture de l'alerte a partir duquel l'alerte a ete lue, pouvant prendre les valeurs suivantes: WEB, APPLI, WEBMOB

☐ Date et heure de lecture de l'alerte sous le format: JJ/MM/AAAA HH:MM:SS

PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU

Quotidienne

XML

Subscribe

If one of the input fields is left blank, the user will be informed.

HUB

A469255

Directory

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a5103\_alg/alertes\_lues/aaaammjj

Identifiant de l'alerte generee

Code canal de lecture de l'alerte a partir duquel l'alerte a ete lue, pouvant prendre les valeurs suivantes: WEB, APPLI, WEBMOB

Date et heure de lecture de l'alerte sous le format: JJ/MM/AAAA HH:MM:SS

Quotidienne

XML

Subscribe

If the entered frequency is larger than the file delivering frequency, the user will be informed and prevented from the subscribe action. For example, if the files are being delivered daily but the user wants them to be transmitted to his project space every hour, it is not possible.

HUB

A469255

×

This subscription cannot be made because the entered frequency is larger than receiving frequency.

Directory

/lake/bddf/a510

Identifiant de l'alerte generee

subscribe

/lake/bddf/a510

Code canal de lecture de l'alerte a partir duquel l'alerte a ete lue, pouvant prendre les valeurs suivantes: WEB, APPLI, WEBMOB

subscribe

/lake/bddf/a510

Date et heure de lecture de l'alerte sous le format: JJ/MM/AAAA HH:MM:SS

subscribe

/lake/bddf/a510

subscribe

/lake/bddf/a510

subscribe

/lake/bddf/a5103\_alg/alertes\_lues/aaaammjj

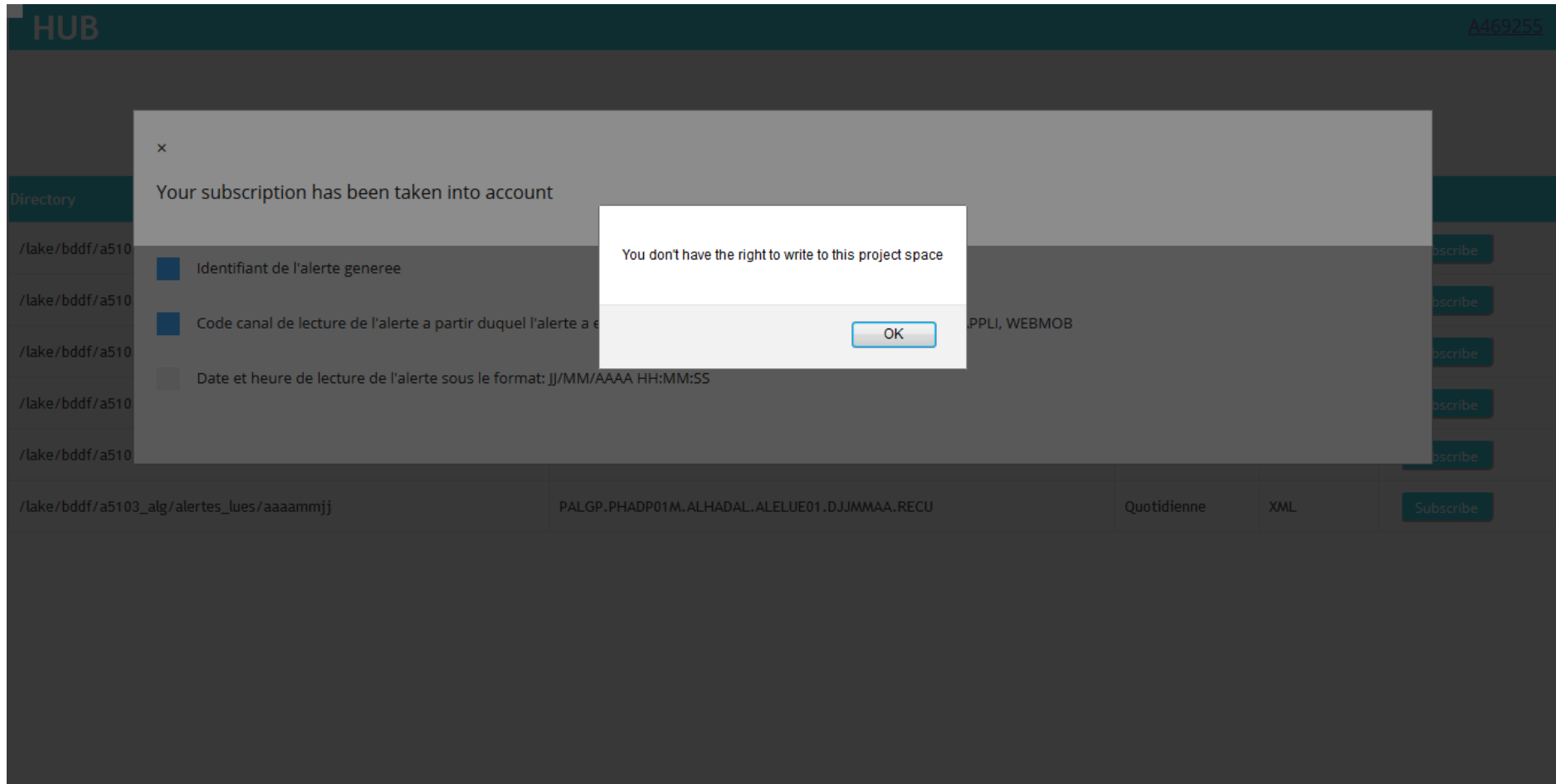
PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU

Quotidienne

XML

Subscribe

User will be prevented from making a subscription to project spaces which he doesn't have the "write" right. For example, user A469255 doesn't belong to cdn group and thus doesn't have the right to write to /project/cdn/dec.





If the subscription details are ok, the subscription will be taken into account. In this example, user A469255 subscribes to the first 2 columns of structure PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU. He wants the files of this structure to be delivered to project space “/project/bddf/hub/storage/test” monthly as csv format, and only wants the most recent files to be transferred.

HUB

A469255

Directory

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a5103\_alg/alertes\_lues/aaaammjj

/project/bddf/hub/storage/te

Quotidienne

csv

☒ Only transfer most recent files

Subscribe

PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU

Identifiant de l'alerte generee

Code canal de lecture de l'alerte a partir duquel l'alerte a ete lue, pouvant prendre les valeurs suivantes: WEB, APPLI, WEBMOB

Date et heure de lecture de l'alerte sous le format: JJ/MM/AAAA HH:MM:SS

PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU

Quotidienne

XML

Subscribe

When a subscription is made, two things happen. First, an oozie workflow is scheduled with the frequency similar the user's input. In our example, the user's input is "Mensuel", thus the oozie workflow is scheduled to be launched every day. Second, all of the details related to this subscription are written to the hub\_registration table in the PostgreSQL database. On the upper figure, we can see that the job ID 000367761807161451589096oozie-oozi-C has been launched. On the lower figure, we can see that this information has been registered.

COORDINATOR

[BDDF] [hub-1.0-SNAPSHOT] [DEV]

SUBMITTER

hubpdadm

STATUS

RUNNING

PROGRESS

0%

FREQUENCY

1 MONTH

NEXT MATERIALIZED TIME

Mon, 27 Aug 2018 17:02:00

ID

0017233-180716145158909-oozie-oozi-C

Coordinator [BDDF] [hub-1.0-SNAPSHOT] [DEV]

Calendar

Actions

Details

Configuration

Log

Definition

Filter results

Succeeded

Running

Error

☐ Day

Warning message

☐ 1-27 Jul 2018 17:02:00

-

← Previous

Next →

Back

```

habilbgd=> select * from hub_registration where coordid='0017233-180716145158909-oozie-oozi-C';
userid |          project          |          registered_structure          | registered_attributes | frequency |
valid |          coordid          | registration_date | cancel_date |
-----+-----+-----+-----+-----+
A469255 | /project/bddf/hub/storage/test | PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMAA.RECU | id_alerte,canal_lecture | Mensuelle |
yes | 0017233-180716145158909-oozie-oozi-C | 2018-07-27T17:02+0200 | 999 |
(1 row)

```






The date today is 20/7/2018. Therefore, when the first oozie workflow has been successfully executed (first materialized time), all of the files with structure PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU that are received between 27/6/2018 and 27/7/2018 are delivered to /project/bddf/hub/storage/test. Since no information is lost after the transmission, the file name will end with 'complete'. On the lower figure, we can see that the data in partition 20170331 is not being transmitted, as it was received on 16/02/2018.

Home

/ lake / bddf / a5103\_alg / alertes\_lues

History

Trash





<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	 <a href="#">↑</a>		hdfs	hdfs	drwx----	July 05, 2018 10:42 AM
<input type="checkbox"/>	 .		hdfs	hdfs	drwx----	July 13, 2018 02:06 PM
<input type="checkbox"/>	 <a href="#">20170331</a>		hdfs	hdfs	drwx----	February 16, 2018 10:17 AM
<input type="checkbox"/>	 <a href="#">20180624</a>		X170439	hdfs	drwx----	July 02, 2018 04:51 PM
<input type="checkbox"/>	 <a href="#">20180713</a>		A469255	hdfs	drwx----	July 13, 2018 02:07 PM

Home

/ project / bddf / hub / storage / test

History

Trash

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	 <a href="#">↑</a>		A469255	hdfs	drwx----	July 20, 2018 01:38 PM
<input type="checkbox"/>	 .		A469255	hdfs	drwx----	July 27, 2018 05:05 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D130718_RECU-494445-complete.csv</a>	515.6 KB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:05 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D240618_RECU-494445-complete.csv</a>	515.6 KB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:05 PM

File delivered date

The date today

All of the input files are of XML format and all of the delivered files will be of csv format.

Download

View file location

Refresh

**Last modified**  
07/13/2018 12:07 PM

**User**  
A469255

**Group**  
hdfs

**Size**  
5.26 MB

/ lake / bddf / a5103\_alg / alertes\_lues / 20180713 / PALGP.PHADP01M.ALHADAL.ALELUE01.D130718.RECU

```
<?xml version="1.0" encoding="UTF-8"?><listeAlertesLues><alerteLue><id_alerte>139891690</id_alerte><canal_lecture>APPLI</canal_lecture><date_heure_lecture_aler
te>24/06/2018 13:19:02</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139466541</id_alerte><canal_lecture>APPLI</canal_lecture><date_heure_lectur
e_alerte>24/06/2018 08:04:57</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>138889355</id_alerte><canal_lecture>WEB</canal_lecture><date_heure_le
cture_alerte>24/06/2018 09:21:44</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139742201</id_alerte><canal_lecture>APPLI</canal_lecture><date_he
ure_lecture_alerte>23/06/2018 17:57:37</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139742217</id_alerte><canal_lecture>APPLI</canal_lecture><d
ate_heure_lecture_alerte>24/06/2018 10:22:54</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139870642</id_alerte><canal_lecture>APPLI</canal_lect
ure><date_heure_lecture_alerte>24/06/2018 10:05:31</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139872230</id_alerte><canal_lecture>WEB</canal_
lecture><date_heure_lecture_alerte>24/06/2018 10:08:00</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139872239</id_alerte><canal_lecture>APPLI</
canal_lecture><date_heure_lecture_alerte>24/06/2018 10:44:48</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139877456</id_alerte><canal_lecture>A
PPLI</canal_lecture><date_heure_lecture_alerte>24/06/2018 15:05:09</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>139466683</id_alerte><canal_lec
ture>APPLI</canal_lecture><date_heure_lecture_alerte>24/06/2018 10:13:01</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>129540403</id_alerte><can
al_lecture>APPLI</canal_lecture><date_heure_lecture_alerte>24/06/2018 10:55:22</date_heure_lecture_alerte></alerteLue><alerteLue><id_alerte>129537216</id_alert
```

Download

View file location

Refresh

**Last modified**  
07/27/2018 3:05 PM

**User**  
hubpdadm

**Group**  
hdfs

**Size**  
515.56 KB

**Mode**  
100600

/ project / bddf / hub / storage / test /

PALGP\_PHADP01M\_ALHADAL\_ALELUE01\_D130718\_RECUI-494445-complete.csv

```
id_alerte;canal_lecture
139891690;APPLI
139466541;APPLI
138889355;WEB
139742201;APPLI
139742217;APPLI
139870642;APPLI
139872230;WEB
139872239;APPLI
139877456;APPLI
139466683;APPLI
129540403;APPLI
129537216;WEB
```

A hive table for PALGP.PHADP01M.ALHADAL.ALELUE01.D130718.RECU is also created:

```
scala> spark.sql("select * from palgp_phadp01m_alhadal_alelue01_d130718_recu").show()
+-----+-----+-----+
|canal_lecture|date_heure_lecture_alerte|id_alerte|
+-----+-----+-----+
|APPLI|24/06/2018 13:19:02|139891690|
|APPLI|24/06/2018 08:04:57|139466541|
|WEB|24/06/2018 09:21:44|138889355|
|APPLI|23/06/2018 17:57:37|139742201|
|APPLI|24/06/2018 10:22:54|139742217|
|APPLI|24/06/2018 10:05:31|139870642|
|WEB|24/06/2018 10:08:00|139872230|
|APPLI|24/06/2018 10:44:48|139872239|
|APPLI|24/06/2018 15:05:09|139877456|
|APPLI|24/06/2018 10:13:01|139466683|
|APPLI|24/06/2018 10:55:22|129540403|
|WEB|24/06/2018 10:37:53|129537216|
|APPLI|23/06/2018 20:38:37|138486490|
|APPLI|23/06/2018 22:21:15|136371447|
|WEBMOB|23/06/2018 22:14:31|137575702|
|WEBMOB|24/06/2018 10:43:50|136466210|
|APPLI|23/06/2018 21:50:34|136625620|
|WEB|24/06/2018 02:55:30|136625841|
|APPLI|23/06/2018 19:01:22|139184318|
|WEB|24/06/2018 09:56:54|136634360|
+-----+-----+-----+
only showing top 20 rows
```

When the user reloads the search results for application a5103, he sees 2 options for PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMAA.RECU: “Modify” and “Unsubscribe”.

HUB

A469255

Directory	File Name	Frequency	File Format	
/lake/bddf/a5103_alg/types_alertes/aaaammjj	PALGP.PHADP01M.ALHADTY.TYPALE01.DJJMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/cat_alertes/aaaammjj	PALGP.PHADP01M.ALHADCT.CATALE01.DJJMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/param_client/aaaammjj	PALGP.PHADP01M.ALHADP*.PARCLI01.DJJMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/alertes/aaaammjj	PALGP.PHADP01M.ALHADA*.ALEGEN01.DJJMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/param_client_supp/aaaammjj	PALGP.PHADP01M.ALHADPS.PARSUP01.DJJMAA.RECU	Quotidienne	XML	<input type="button" value="Subscribe"/>
/lake/bddf/a5103_alg/alertes_lues/aaaammjj	PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMAA.RECU	Quotidienne	XML	<input type="button" value="Modify"/> <input type="button" value="Unsubscribe"/>

When the same user (A469255) wants to subscribe the same project space (/project/bddf/hub/storage/test) to the same file structure (PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU) but to different sets of columns two things will happen. First, the existing subscription with the same aforementioned information will be cancelled. That is, the validity of this subscription in the hub\_registration table will be set to “no” and the oozie coordinator that runs this subscription will be killed. Second, the new subscription will be taken into account. That is, the details of the new subscription will be written to the hub\_registration table, and a new oozie application will be scheduled.

HUB

A469255

Directory

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a510

/lake/bddf/a5103\_alg/alertes\_lues/aaaammjj

x

/project/bddf/hub/storage/hub

Mensuelle

CSV

☐ Only transfer most recent files

Subscribe

PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU

■

Identifiant de l'alerte generee

■

Code canal de lecture de l'alerte a partir duquel l'alerte a ete lue, pouvant prendre les valeurs suivantes: WEB, APPLI, WEBMOB

■

Date et heure de lecture de l'alerte sous le format: JJ/MM/AAAA HH:MM:SS

/lake/bddf/a5103_alg/alertes_lues/aaaammjj	PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU	Quotidienne	XML	<div>Modify</div> <div>Unsubscribe</div>
--	--	-------------	-----	--

On the upper left figure, we can see that the job ID 00036776180716145158909oozie-oozi-C has been killed. On the upper right figure, we can see the job ID 0003692-180716145158909-oozie-oozi-C. And on the lower figure, we can see that this new information has been registered to PostgreSQL DB.

#### COORDINATOR

[BDDF] [hub-1.0-SNAPSHOT]  
[DEV]

#### SUBMITTER

hubpdadm

#### STATUS

**KILLED**

#### PROGRESS

0%

#### FREQUENCY

1 MONTH

#### NEXT MATERIALIZED TIME

Mon, 27 Aug 2018 17:02:00

#### ID

0017233-180716145158909-  
oozie-oozi-C

#### COORDINATOR

[BDDF] [hub-1.0-SNAPSHOT]  
[DEV]

#### SUBMITTER

hubpdadm

#### STATUS

**RUNNING**

#### PROGRESS

0%

#### FREQUENCY

1 MONTH

#### NEXT MATERIALIZED TIME

Mon, 27 Aug 2018 17:13:00

#### ID

0017241-180716145158909-  
oozie-oozi-C

#### Coordinator [BDDF] [hub-1.0-SNAPSHOT] [DEV]

Calendar

Actions

Details

Configuration

Log

Definition

Filter results

☐

Day

Warning message

☐

1-27 Jul 2018 17:13:00

-

Back

```

hablbgd=> select * from hub_registration;
userid | project | frequency | valid | coordid | registered_structure | registration_date | registered_attributes
-----+-----+-----+-----+-----+-----+-----+-----
A469255 | /project/bddf/hub/storage/test | Mensuelle | no | 0003677-180716145158909-oozie-oozi-C | PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMAA.RECU | 2018-07-20T14:00+0200 | id_alerte,canal_lecture
A469255 | /project/bddf/hub/storage/test | Mensuelle | yes | 0003692-180716145158909-oozie-oozi-C | PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMAA.RECU | 2018-07-20T14:18+0200 | id_alerte,canal_lecture,date_heure_lecture_a
(2 rows)

```










This time, since the user wants to transfer everything (not only most recent files), the file PALGP\_PHADP01M\_ALHADAL\_ALELUE01\_D033117\_RECUI will be transmitted as well.

Home

/ project / bddf / hub / storage / test

History

Trash

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	 <a href="#">↑</a>		A469255	hdfs	drwx----	July 20, 2018 01:38 PM
<input type="checkbox"/>	 .		A469255	hdfs	drwx----	July 27, 2018 05:15 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D033117_RECUI-93-complete.csv</a>	146 bytes	hubpdadm	hdfs	-rw-----	July 27, 2018 05:15 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D130718_RECUI-1163765-complete.csv</a>	1.1 MB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:15 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D130718_RECUI-494445-complete.csv</a>	515.6 KB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:05 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D240618_RECUI-1163765-complete.csv</a>	1.1 MB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:15 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D240618_RECUI-494445-complete.csv</a>	515.6 KB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:05 PM

It also works when input is a json file and output is an xml file:

Edit file

Download

View file location

Refresh

/ lake / bddf / a1802\_odp / hprddgaparse / 20180306 / file-stream-2-evening

```
{ "timestamp_da": "2018-03-06T12:50:00.000+01:00", "PROGRAM": "a1802_odp", "MESSAGE": "Evenement de test ORCHESTRA bddf" }
{ "timestamp_da": "2018-03-06T12:50:00.000+02:00", "PROGRAM": "a1802_odp", "MESSAGE": "Evenement de test ORCHESTRA bddf" }
```

All files of structure file-stream-\$-morning, file-stream-\$-evening in /lake/bddf/a1802\_odp/hprddgaparse/ have been transmitted to /project/bddf/hub/storage/test

COORDINATOR

[BDDF] [hub-1.0-SNAPSHOT] [DEV]

SUBMITTER

hubpdadm

STATUS

RUNNING

PROGRESS

0%

FREQUENCY

1 DAY

NEXT MATERIALIZED TIME

Sat, 28 Jul 2018 17:26:00

ID

0017253-180716145158909-oozie-oozi-C

Coordinator [BDDF] [hub-1.0-SNAPSHOT] [DEV]

Calendar

Actions

Details

Configuration

Log

Definition

Filter results

Succeeded Running Error

Day

Warning message

1-27 Jul 2018 17:26:00










-

Back

← Previous

Next →

42

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	 <a href="#">↑</a>		A469255	hdfs	drwx----	July 20, 2018 01:38 PM
<input type="checkbox"/>	 .		A469255	hdfs	drwx----	July 27, 2018 05:33 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D033117_RECUI-93-complete.csv</a>	146 bytes	hubpdadm	hdfs	-rw-----	July 27, 2018 05:15 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D130718_RECUI-1163765-complete.csv</a>	1.1 MB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:15 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D130718_RECUI-494445-complete.csv</a>	515.6 KB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:05 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D240618_RECUI-1163765-complete.csv</a>	1.1 MB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:15 PM
<input type="checkbox"/>	 <a href="#">PALGP_PHADP01M_ALHADAL_ALELUE01_D240618_RECUI-494445-complete.csv</a>	515.6 KB	hubpdadm	hdfs	-rw-----	July 27, 2018 05:05 PM
<input type="checkbox"/>	 <a href="#">file_stream_2_evening-144-complete.xml</a>	385 bytes	hubpdadm	hdfs	-rw-----	July 27, 2018 05:33 PM
<input type="checkbox"/>	 <a href="#">file_stream_3_evening-72-complete.xml</a>	200 bytes	hubpdadm	hdfs	-rw-----	July 27, 2018 05:33 PM

Edit file

/ project / bddf / hub / storage / test / **file\_stream\_3\_evening-72-complete.xml**

Download

View file location

Refresh

Last modified  
07/27/2018 3:33 PM

```
<ROWS>
  <ROW>
    <timestamp_da>2018-03-06T12:50:00.000+00:00</timestamp_da>
    <PROGRAM>a1802_odp</PROGRAM>
    <MESSAGE>Evenement de test ORCHESTRA bddf</MESSAGE>
  </ROW>
</ROWS>
```

Similarly, when input is CCB and output is json:

/ lake / bddf / a0877\_urta / urta / test / data / 20180627 / **test\_data.txt**

A0877300030099100120170523152104A131785308VCB 30800008002017052310201705230353000300991000000009910006751158700000000010000R2DEUR30800008 0 00000700000

A0877300030099100120170523152138A131785308VCB 30800009002017052310201705230353000300991000000014060003416285500000000010000I2CEUR30800008 0 000048  
0000000000000000€0

A0877300030099100120170523180425A131785308VCB 3080001200201705231020170523035300030099100000009910005025273400000000000160J2EUR30800012 0 000050  
0000000000000000€0

A0877300030099100120170523180506A131785308VCB 3080001300201705231020170523035300030099100000009910005030229900000000000160A2CEUR30800012 0 000050  
0000000000000000€0

A0877300030099900120170523163747A454424310VCB 31000001002017052310201705230563000301010000000010100005100373000000000010547702EUR31000001 0 000051  
0000000000000000€0

```

* ==> FLUX 10
01  STR10-NC.
    10 WW00-IDPIDA-B
        15 WW00-CDAPIN-B      PIC X(5).
        15 WW00-IDELSS-B      PIC X(10).
        15 WW00-IDCADI-B      PIC X(3).
        15 WW00-DASAO9-B      PIC 9(8).
        15 WW00-HESAO2-B      PIC 9(6).
        15 WW00-IDAGNT-B      PIC X(7).
        15 WW00-NUMTER-B      PIC X(3).
        15 WW00-CDTRS1-B      PIC X(5).
        15 WW00-NUSQA1-B      PIC 9(8).
        15 WW00-NUOR02-B      PIC 9(2).

    10 WW00-DATRTJ-B          PIC 9(8).
    10 WW00-CDSTRJ-B          PIC X(2).

    10 WN10-DATEJO-NC          PIC 9(8).
    10 WN10-ARTICLE-NC          PIC 9(3).
    10 WN10-ENTGES-NC          PIC X(10).
    10 WN10-NUMCPT-NC          PIC X(22).
    10 WN10-MONTANT-NC          PIC S9(16).

```

hubpdadm

## STATUS

RUNNING

## PROGRESS

0%

## FREQUENCY

1 DAY

## NEXT MATERIALIZED TIME

Sat, 28 Jul 2018 17:40:00

## ID

0017265-180716145158909-  
oozie-oozi-C

Export

Download

View file location

Refresh

## Last modified

07/27/2018 3:42 PM

## User

hubpdadm

## Group

hdfs

## Size

5.6 KB

## Mode

100600

☐ Day

Warning message

☐

1-27 Jul 2018 17:40:00

-

← Previous

Next →

Back

/ project / bddf / hub / storage / test / test\_data\_txt-1490-complete.json

```
{
  "WW00_CDAPIN_B": "A0877",
  "WW00_IDELSS_B": "3000300991",
  "WW00_IDCADI_B": "001",
  "WW00_DASAO9_B": "20170523",
  "WW00_HESAO2_B": "152104",
  "WW00_IDAGNT_B": "A131785",
  "WW00_NUMTER_B": "308",
  "WW00_CDTRS1_B": "VCB",
  "WW00_NUSQA1_B": "30800008",
  "WW00_NUOR02_B": "00",
  "WW00_DATRTJ_B": "20170523",
  "WW00_CDS_TRJ_B": "10",
  "WW10_DATEJO_NC": "20170523",
  "WW10_ARTICLE_NC": "035",
  "WW10_ENTGES_NC": "3000300991",
  "WW10_NUMCPT_NC": "0000000099100067511587",
  "WW10_MONTANT_NC": "00000000010000R",
  "WW10_NBDEC_NC": "2",
  "WW10_SENS_NC": "D",
  "WW10_DEVISE_NC": "EUR",
  "WW10_LIEN_ECR_NC": "30800008",
  "WW10_TOP_STATUT_ECR_NC": " ",
  "WW10_COD_SUP_CPT_NC": "0",
  "WW10_IND_DEC_BAN_NC": " ",
  "WW10_COD_EXO_NC": "0",
  "WW10_COD_UNIQ_NC": "00007",
  "WW10_GUI_CAISSE_NC": ""
}

{
  "WW00_CDAPIN_B": "A0877",
  "WW00_IDELSS_B": "3000300991",
  "WW00_IDCADI_B": "001",
  "WW00_DASAO9_B": "20170523",
  "WW00_HESAO2_B": "152138",
  "WW00_IDAGNT_B": "A131785",
  "WW00_NUMTER_B": "308",
  "WW00_CDTRS1_B": "VCB",
  "WW00_NUSQA1_B": "30800009",
  "WW00_NUOR02_B": "00",
  "WW00_DATRTJ_B": "20170523",
  "WW00_CDS_TRJ_B": "10",
  "WW10_DATEJO_NC": "20170523",
  "WW10_ARTICLE_NC": "035",
  "WW10_ENTGES_NC": "3000300991",
  "WW10_NUMCPT_NC": "0000000140600034162855",
  "WW10_MONTANT_NC": "00000000010000I",
  "WW10_NBDEC_NC": "2",
  "WW10_SENS_NC": "C",
  "WW10_DEVISE_NC": "EUR",
  "WW10_LIEN_ECR_NC": "30800008",
  "WW10_TOP_STATUT_ECR_NC": " ",
  "WW10_COD_SUP_CPT_NC": "0",
  "WW10_IND_DEC_BAN_NC": " ",
  "WW10_COD_EXO_NC": "0",
  "WW10_COD_UNIQ_NC": "00048",
  "WW10_GUI_CAISSE_NC": ""
}

{
  "WW00_CDAPIN_B": "A0877",
  "WW00_IDELSS_B": "3000300991",
  "WW00_IDCADI_B": "001",
  "WW00_DASAO9_B": "20170523",
  "WW00_HESAO2_B": "180425",
  "WW00_IDAGNT_B": "A131785",
  "WW00_NUMTER_B": "308",
  "WW00_CDTRS1_B": "VCB",
  "WW00_NUSQA1_B": "30800012",
  "WW00_NUOR02_B": "00",
  "WW00_DATRTJ_B": "20170523",
  "WW00_CDS_TRJ_B": "10",
  "WW10_DATEJO_NC": "20170523",
  "WW10_ARTICLE_NC": "035",
  "WW10_ENTGES_NC": "3000300991",
  "WW10_NUMCPT_NC": "0000000099100050252734",
  "WW10_MONTANT_NC": "00000000000160J",
  "WW10_NBDEC_NC": "2",
  "WW10_SENS_NC": "D",
  "WW10_DEVISE_NC": "EUR",
  "WW10_LIEN_ECR_NC": "30800012",
  "WW10_TOP_STATUT_ECR_NC": " ",
  "WW10_COD_SUP_CPT_NC": "0",
  "WW10_IND_DEC_BAN_NC": " ",
  "WW10_COD_EXO_NC": "0",
  "WW10_COD_UNIQ_NC": "00050",
  "WW10_GUI_CAISSE_NC": ""
}
```

When input is of type csv/txt and output is of type json:

[Home](#) / [lake](#) / [bddf](#) / [a0877\\_urta](#) / [ref\\_transactions](#) / [ref\\_transaction.csv](#)Page  to  of 5 [⏪](#) [⏴](#) [⏵](#) [⏩](#)

```
Code transaction;Libellé;Description ;"Début PROD"
ASA;Adhésion aScopAr;Permet de saisir les adhésions ASCOPAR souscrites par les clients titulaires d'un compte à vue (code produit 10, 11, 50, 51);12/05/2004
RCA;Affectation caisse;"Permet de :          - Affecter une caisse euro à un agent quin'en détient pas          - Désaffecter la caisse d'un agent          - Transfère
r l'encaisse euro d'une caisse à une autre";01/07/2004
AFD;Affectation caisse devises;"Permet :          - D'affecter ou de désaffecter une caisse devise à un agent          - De transférer la caisse devises d'un agent d
ans l'un des guichets du groupe";01/07/2004
CPT;Alimentation compte de placement;permet d'effectuer des virements sur ou à partir d'un compte de placement en relation avec un compte vue appartenant au même
tiers que ce compte de placement (donc pas de déclaration balance des paiements). La transaction permet 1 débit et 1 crédit.;15/11/2006
DEL;Annonce de délestage;Utilisée pour délester des fonds d'une caisse automatique vers un coffre tire-lire;11/02/2004
AVI;Annonce de vidage;Permet,au personnel autorisé, d'annoncer un vidage d'une caisse automatique pour le jour même ou le lendemain,,(le vidage ne pourra avoir l
ieu qu'une 1/2 heure après la saisie de la transaction AVI pour un vidage le jour même);14/09/2005
AVA;Annulation virement agendé;Permet d'annuler un virement agendé;12/05/2004
```

[Download](#)[View file location](#)[Refresh](#)

**Last modified**  
07/27/2018 3:47 PM

**User**  
hubpdadm

**Group**  
hdfs

**Size**  
25.4 KB

**Mode**

```
{
  "Code_transaction": "Code transaction",
  "Libelle": "Libellé",
  "Description": "Description ",
  "Debut_PROD": "Debut_PROD"
}
{
  "Code_transaction": "ASA",
  "Libelle": "Adhésion aScopAr",
  "Description": "Permet de saisir les adhésions ASCOPAR souscrites par les clients titulai
res d'un compte à vue (code produit 10, 11, 50, 51)",
  "Debut_PROD": "Debut_PROD"
}
{
  "Code_transaction": "RCA",
  "Libelle": "Affectation caisse",
  "Description": "\"Permet de :          - Affecter une caisse euro à un agent quin'en déti
ent pas          - Désaffecter la caisse d'un agent          - Transférer l'encaisse euro d'une caisse à une autre\"",
  "Debut_PROD": "Debut_PROD"
}
{
  "Code_transaction": "AFD",
  "Libelle": "Affectation caisse devises",
  "Description": "\"Permet :          - D'affecter ou de désaffecter une caisse dev
ise à un agent          - De transférer la caisse devises d'un agent dans l'un des guichets du groupe\"",
  "Debut_PROD": "Debut_PROD"
}
{
  "Code_transaction": "CPT",
  "Libelle": "Alimentation compte de placement",
  "Description": "permet d'effectuer des virements sur ou à partir d'un com
pte de placement en relation avec un compte vue appartenant au même tiers que ce compte de placement (donc pas de déclaration balance des paiem
ents). La transaction permet 1 débit et 1 crédit.",
  "Debut_PROD": "Debut_PROD"
}
{
  "Code_transaction": "DEL",
  "Libelle": "Annonce de délestage",
  "Description": "Utilisée pour délester des fonds d'une caisse automatique vers un cof
fre tire-lire",
  "Debut_PROD": "Debut_PROD"
}
{
  "Code_transaction": "AVI",
  "Libelle": "Annonce de vidage",
  "Description": "Permet,au personnel autorisé, d'annoncer un vidage d'une caisse automati
que pour le jour même ou le lendemain,,(le vidage ne pourra avoir lieu qu'une 1/2 heure après la saisie de la transaction AVI pour un vidage
le jour même)",
  "Debut_PROD": "Debut_PROD"
}
```

When the user unsubscribes from a structure, the existing subscription for this structure is cancelled. That is, the validity of this subscription in the hub\_registration table will be set to “no” and the oozie coordinator that runs this subscription will be killed. On the upper figure, we can see that the oozir job ID 0003692-180716145158909-oozie-oozi-C has been killed. On the lower figure, we can see that this new information has been registered.

**COORDINATOR**

[BDDF] [hub-1.0-SNAPSHOT] [DEV]

**SUBMITTER**

hubpdadm

**STATUS**

**KILLED**

**PROGRESS**

0%

**FREQUENCY**

1 MONTH

**NEXT MATERIALIZED TIME**

Mon, 27 Aug 2018 17:13:00

**ID**

0017241-180716145158909-oozie-oozi-C

**Coordinator [BDDF] [hub-1.0-SNAPSHOT] [DEV]**

Calendar Actions Details Configuration Log Definition

Filter results

Succeeded Running Error

☐ Day Warning message

☐ 1-27 Jul 2018 17:13:00 -

← Previous Next →

Back

```

habilbgd=> select * from hub_registration where coordid='0017241-180716145158909-oozie-oozi-C';
  userid |      project      | registered_structure | registered_attributes
-----+-----+-----+-----+-----+-----+-----+-----
  A469255 | /project/bddf/hub/storage/test | PALGP.PHADP01M.ALHADAL.ALELUE01.DJJMMAA.RECU | id_alerte,canal_lecture,date_heure_lecture_alerte | Mensuelle | no      | 0017241-180716145158909-oozie-oozi-C | 2018-07-27T17:13+0200 | 2018-07-27T17:50+0200
(1 row)

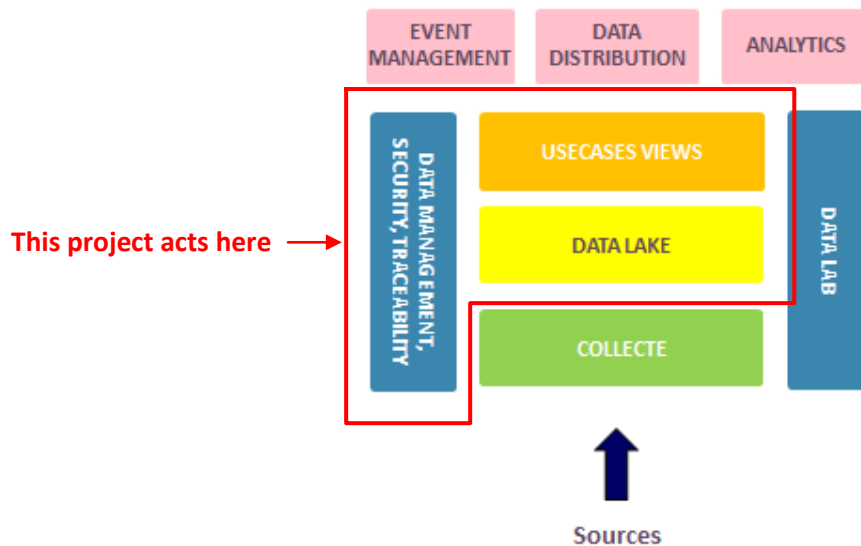
```



## Chapter 5

### Conclusion

This project consists of 20% Web App development, 20% architecture integration (designing a solution that fits the current big data architecture at SG, dealing with security and permission related issues around the HDFS) and 60% big data development (using spark and oozie), as shown in figure 4.



**Figure 4 - Architecture of the big data platform functions**

The biggest challenges of this project are related to architecture integration, particularly to understand the role of LDAP and Ranger in the HDFS security checking; to get the permission to use LDAP, Ranger, PostgreSQL, to create a new HTTP server on the HDFS; and to validate the design of the application. Time constraint is something important to be taken into account, as the above processes usually consist of many phases and require many architecture validations, especially for a newly-launch project as this one.

The project has successfully resolved two issues described in session 1.1. In particular, it has standardized the way to deliver data for end-usage by doing two things: accepting parameters needed for files transmission via an interface, and picking the right treatment method for the transmission using these parameters. It has also prevented the risk of having a lot of people accessing data that they are not supposed to access to by allowing only administrative users to access to sources.

Overall, the project has met all of the technical requirements, as shown in chapter 4. It has also attracted many developers and will get to be further developed in the future. The main future work for this project will be related to hive table pretreatments before writing them to project space. In particular, these treatments are checking the quality of the data in the hive tables (for example, if they contain any outlier), and providing the possibility to join several hive tables on a common key.

## References

1. REST APIs for Policy Management.  
<https://cwiki.apache.org/confluence/display/RANGER/REST+APIs+for+Policy+Management>
2. Common Gateway Interface. [https://fr.wikipedia.org/wiki/Common\\_Gateway\\_Interface](https://fr.wikipedia.org/wiki/Common_Gateway_Interface)
3. HDFS Architecture Guide. [https://hadoop.apache.org/docs/r1.2.1/hdfs\\_design.html](https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html)
4. Apache Oozie workflow scheduler for Hadoop. <http://oozie.apache.org/>
5. LDAP (Lightweight Directory Access Protocol).  
<https://searchmobilecomputing.techtarget.com/definition/LDAP>
6. Apache Ranger. <https://ranger.apache.org/>
7. Apache Maven Project. <https://maven.apache.org/>

## Appendix 1 – Manually compile HTTP Server

```
[A469255@dhadlx112 sources]$ tar xvfz /home/x120832/apr-1.6.2.tar.gz
[A469255@dhadlx112 sources]$ cd apr-1.6.2/
[A469255@dhadlx112 apr-1.6.2]$ ./configure --prefix=/applis/hadd/produits/glpi/apr --exec-
prefix=/applis/hadd/produits/glpi/apr
[A469255@dhadlx112 apr-1.6.2]$ make
[A469255@dhadlx112 apr-1.6.2]$ make install
[A469255@dhadlx112 sources]$ tar xzvf /home/x120832/apr-util-1.6.0.tar.gz
[A469255@dhadlx112 sources]$ cd apr-util-1.6.0/
[A469255@dhadlx112 apr-util-1.6.0]$ ./configure --prefix=/applis/hadd/produits/glpi/apr-util --exec-
prefix=/applis/hadd/produits/glpi/apr-util --with-apr=/applis/hadd/produits/glpi/apr
[A469255@dhadlx112 apr-util-1.6.0]$ make
[A469255@dhadlx112 apr-util-1.6.0]$ make install
[A469255@dhadlx112 sources]$ tar xvfz /home/x120832/expat-2.2.1.tar.bz2
[A469255@dhadlx112 sources]$ cd expat-2.2.1/
[A469255@dhadlx112 expat-2.2.1]$ ./configure --prefix=/applis/hadd/produits/glpi/expat --exec-
prefix=/applis/hadd/produits/glpi/expat
[A469255@dhadlx112 expat-2.2.1]$ make
[A469255@dhadlx112 expat-2.2.1]$ make install
[A469255@dhadlx112 sources]$ tar xvfz /home/x120832/pcre-8.41.tar.gz
[A469255@dhadlx112 sources]$ cd pcre-8.41/
[A469255@dhadlx112 pcre-8.41]$ ./configure --prefix=/applis/hadd/produits/glpi/pcre --exec-
prefix=/applis/hadd/produits/glpi/pcre
[A469255@dhadlx112 pcre-8.41]$ make
[A469255@dhadlx112 pcre-8.41]$ make install
[A469255@dhadlx112 sources]$ tar xvfz /home/x120832/httpd-2.4.26.tar.gz
[A469255@dhadlx112 sources]$ cd httpd-2.4.26/
[A469255@dhadlx112 httpd-2.4.26]$ ./configure --prefix=/applis/hadd/produits/glpi/httpd --exec-
prefix=/applis/hadd/produits/glpi/httpd --with-pcre=/applis/hadd/produits/glpi/pcre/bin/pcre-config
--with-apr=/applis/hadd/produits/glpi/apr --with-apr-util=/applis/hadd/produits/glpi/apr-util --with-
expat=/applis/hadd/produits/glpi/expat
[A469255@dhadlx112 httpd-2.4.26]$ make
[A469255@dhadlx112 httpd-2.4.26]$ make install
```