Summer Student Report
Jessie Abramson

0- Introduction

The AV Workflow is web application which allows cern users to publish, update and delete videos from cds. During this internship I implemented the backend of the new version of the AV Worklow in python.

1- Why did we do a new workflow?

The old AV-Workflow was only accepting videos in the ".mov" format for the master video. Users had to be in existing communities to be able to submit a video to cds. The users also had to use Final Cut Server in order to publish a video. Those are limitations that the new av workflow removed.
In addition to that it's difficult to implement new features and update the current workflow as it has been made in Ruby which is not a language known by many developers in the department, and the workflow had a very limited documentation.

2- The AV-Workflow

The Workflow consists of multiple steps. The user first has to type some metadata about the project. A project record is then created in the database. He then has to choose an ingestion method, and to type some metadata about the clip. The form to enter the clip metadata is prefilled with information found in the video file and with the metadata from the project. At that point a proxy video is created by transcoding the master with a specific proxy preset. This allows the user to see the clip before submitting it to cds. Finally the user can publish the clip to cds, this involves transcoding the master with multiple adequate presets and sending requests to cds.
In the rest of the report I give more details about some specific parts of the process.

3- Ingestion

The web offer multiple ways to ingest a video. The simplest one is to copy the master file to the master folder, and then give the name of the file to the webapp. After the webapp rename the file and proceed to the transcoding of the proxy video.
The user can enter a link to a video and the server will then download the video from the address. This ingest method requires to change the url for dropbox and for box.cern.ch, as they are not direct links to the video file. This is done with two regular expressions to know which modifications should be made to the link.
The user can also choose to use Vidyo to ingest videos. Vidyo is platform used for video conferences. Vidyo uses a SOAP api. This ingest method consists of a single call, with the

username as a parameter. This call returns the list of the videos recorded by the user and their details in a xml. The title of the video, and the name of file are extracted from the result of the request. The video file is then copied to the master folder.

4- Transcoding

The transcoding can be done by two different software: Episode and Sorenson.
To transcode using Episode the server first has to use ssh to remote login to a server running an episode server. A command is then ran to start the transcoding process. This command precise what encoding preset should be used by giving the path to an epitask file.
The tracking of the transcoding is done in a similar way. After using ssh and running a command on the server, the server outputs some information about two tasks. The first task is the transcoding and the second one correspond to moving the file resulting from the transcoding to the output folder. In the output the information about the two tasks is separated by a new line and the properties are separated by “|”. To parse this output I am first splitting the string in two. Then three regular expressions are used to extract the progress, status and message of the two tasks.
Starting the transcoding with the sorenson engine is done by sending a POST request to the squeeze server with a xml attached as data. The xml contains information about the destination, the source and a preset Id. A preset Id points to a preset stored on the server which specifies how the transcoding should be done. Obtaining an update on the status of the transcoding job is done by sending a GET request to the server, and then parsing xml sent back by the server. The transcoding with the sorenson engine was not fully implemented during my internship.

5- Submission to cds

While doing the transcoding, the web application sends multiple requests to cds to create and update records. Each request to cds is sent with an xml and a callback URL. When cds finishes to process a request, a call is made to AV-Workflow server. This call contains a json which holds the result of the request and the cds id of the record.
The first request to cds create a record for the project on the cds servers. The xml sent with this request is mostly made of the metadata typed by the user. Another request is sent to create a clip record, this xml does not contains the slaves as they have not been generated yet. Once the transcoding of a slave is finished a new request is sent to cds with a correct flag. The xml of this request only contains the id of the records and a list of all the slaves, thumbnails, poster frames and the master of the clip.

Conclusion

The new AV-Workflow webapp solves most of the limitations that the old workflow had. In addition to that it is implemented in python which will make it easier to maintain. Finally the demo of the first beta done at the end of august was successful and the web app is dues to be released in october.