**Documentation for the procedure to retrieve Transaction ID (Online/ Offline Mode)**

**Problem Statement: Undefined transaction id field faced in the front-end scripting in “printerstate.js”.**

**Approach 1**: Writing jQuery AJAX calls to get the transaction id generated for every print command issued and then bind it with the knockout js observable.

**Issue faced** : Scope of the json data got by the success function of AJAX call.

**Approach 2:** Querying the metadata.yaml file to retrieve the transaction id.

**Issue faced** : None.

**APPROACH 2**

Files edited:

(A) printerstate.js | Location: OctoPuppet/src/octoprint/static/js/app/viewmodels

(B) state.jinja2 | Location: OctoPuppet/src/octoprint/templates/sidebar

(C) standard.py | Locations: OctoPuppet/src/octoprint/printer

1. **printerstate.js:**
2. self.\_processProgressData(data):

This function is a knockout observable, it subscribes a computable knockout that processes the data. The function “\_processProgressData” accepts a parameter data that is fed to it through the server side python script and sets the value of the “trans\_id” knockout.

The data parameter accepted by this function holds the data returned after a print command is issued successfully and returns the printTime, printTimeLeft, printTimeLeftOrigin and the transaction id associated with it.

1. self.transId():

This function is a knockout observable that gets set by the function above and is used by the “self.transIdString()”.

1. self.transIdString():

This is a pureComputed knockout function that is used to bind the transaction id to the user interface displayed on the monitors at FabLab. It is dynamically updated using knockout data binds.

1. **state.jinja2:**
2. The data bind for the transaction id in this file is done using the self.transIdString() pureComputed knockout function.

**(C) standard.py:**

1. self.\_stateMonitor.reset():

An additional parameter, “trans\_id” to reset the Transaction Id is declared as an element for the data returned by “progress”.

1. def get\_transID():

This function retrieves the transaction id for a particular print command from .metadata.yaml located at .octoprint/uploads based on the filename uploaded. This is achieved in the following steps:

1. First, the self.\_fileManager.path\_on\_disk() function is called to get the location of the file uploaded and stored as “path\_on\_disk”.
2. Then, the self.\_fileManager.get\_metadata() function is called to get the data related to the current file uploaded and is stored in a variable fileData. This function accepts the location of the file as a parameter.
3. Finally, the data acquired by the above function is traversed to fetch the transaction id and is stored in the transId variable.

The function returns the transaction id.

(3) def \_updateProgressData(self, completion=None, filepos=None, printTime=None,

printTimeLeft=None, transId=None):

This function is used to update the progress data for an issued print command, it keeps updating the data sent to the knockout observable to bind it to be displayed on the User

Interface. This function calls the self.\_stateMonitor.set\_progress() and passes the

transaction id as a parameter that is returned as a dictionary response.

(4) def \_updateProgressDataCallback():

The above function returns the updated progress data along with the

Transaction id as a dictionary. This function first checks if a connection is established, if not

then the transaction id and other data parameters are set to none else the transaction id is retrieved by calling the get\_transID() function. This transaction id is constantly updated after the first print command is issued to make sure the transaction id is fetched and provided to the front-end scripting functions to bind it for display.

(5) def \_setJobData():

The \_setJobData() function first scans the SD storage to find the file uploaded using one of it

parameters; filename. This function calls its internal method \_stateMonitor.set\_job\_data to set

the initial data parameters for a print job including the transaction id. It calls another method;

\_filemanager.get\_metadata() which accepts the filename, the file’s location as parameters

and queries the metadata.yaml file to retrieve the print job data including the transaction id

and stores it in a variable “filedata”. The transaction id from this filedata is accessed by simply traversing the file and pulling out the data value of the “trans\_id” field.

This function then passess this data to the \_stateMonitor.set\_job\_data() function.

(6) def on\_comm\_print\_job\_done():

This function mainly places a call to the .\_updateProgressData() so as to update the data related

to a particular print command issued, this calling updates the progress data for a print

command after its command. The data parameters passed to the .\_updataProgressData()

includes the transaction id retrieved using the get\_transID() function listed above.

NOTE: The .\_updateProgressData() function is called multiple time by various other functions in the file to make sure the data parameters related to a particular print job are constantly up-to-date.