Technical Design for the Registration Packet Status Reader

# **Functional Background**

Registration client application will save the each individual registration details as form as packet in the local disk and same thing should be uploaded successfully to the desired location in the server, where the registration processor has to pick up and process the packets. After that the registration client will be ***periodically/manually check the status*** of the successfully uploaded packets. This will help the Registration client application to ***archives or deletion*** of the packets based on the desired status.

The **target users** are

* Server application, which will provide a Registration Status API to get current status of packets.
* Client application, which will upload the packets to the desired location where the registration processor has to pick up and process. Post uploading the packets, will invoke the Registration Status API to get the status from the server. Either Manual / batch process would initiate this transaction.

The key **requirements** are

* Get registration packet status based on id[s].
* Call rest API to get the enrollment statuses for list of enrollment ids.
* Update registration packet status in the table.
* Display the packet status in the UI if request raised from UI.
* There are two actors applicable to trigger this request.
  + Manual trigger from the Registration client application.
  + Batch trigger from the Registration client application.

The key **non-functional requirements** are

* Connectivity:
  + Should able to communicate to the configured URL with proper authentication.
  + The http read timeout parameter to be explicitly set, if client unable to connect to the REST service.
  + Connectivity should happen through SSL mode. The respective key to be loaded during the call.
* Authentication:
  + While connecting to the server, user authentication is required to authenticate by providing the valid credentials.
  + Invoke the Oauth service to get the ‘Access token’ and pass it along with the request to authenticate the request by the server.
* Database:
  + After getting the status, update each record WRT status.
  + All connection should be closed once db process completed.

# **Technical Approach**

The key solution considerations are -

* Create ***RegistrationPacketStatusService*** and create DTO and DAO layer for status updation.
  + Get list of registration packet by status [Status: POST SYNCHED].
  + Call the Rest API to fetch the status of the list of packet ids.
  + After getting the proper response, if success update the packet status to the

“registration” and “registration\_transactions” tables with desired status.

* Handle exceptions in using custom Exception handler and send correct response to client.

UI

Create the proper alert success/error to intimate the user.

Apply the below common criteria

* Audit
* Log
* Java Documentation
* Junit

**Classes**:

**Controller**:

RegPacketStatusController

**Service**:

RegPacketStatusService

**DTO**:

RegPacketStatusDTO

**DAO**:

RegPacketStatusDAO/Impl

**DB scripts:**



List of Status provided by the Registration-Processor:

|  |  |
| --- | --- |
|  | Rejected - virus found |
|  | Rejected – unable to decrypt the packet. |
|  | Rejected – duplicate packet. |
|  | Rejected – other reason |
|  | UIN Created. |
|  | Reached landing zone. |
|  |  |

Class Diagram:

<https://github.com/mosip/mosip/blob/DEV/design/registration/_images/_class_diagram/registration-packetstatusreader-classDiagram.png>

Sequence Diagram:

<https://github.com/mosip/mosip/blob/DEV/design/registration/_images/_sequence_diagram/registration-packetstatusreader-sequenceDiagram.png>

Response:

Service class provide the list of success / failure detail to the invoking client program.

Client class [Controller] parse the ResponseDTO and frame the following message to be displayed to the UI if manual request triggered by the user.

UI Display:

|  |  |  |
| --- | --- | --- |
| **Enrollment ID** | **Status from Server** | **Additional Comments** |
|  |  |  |
|  |  |  |
|  |  |  |

**Request and Response [RegistrationStatus REST Service]:**

Provided the request and response of the ‘registrationstatus’ service. This service would accept the single / multiple enrollment id and respond with the respective status.

1. <https://github.com/mosip/mosip/blob/DEV/design/registration/_images/registration-packetstatusreader_restrequest.png>
2. <https://github.com/mosip/mosip/blob/DEV/design/registration/_images/registration-packetstatusreader_restresponse.png>