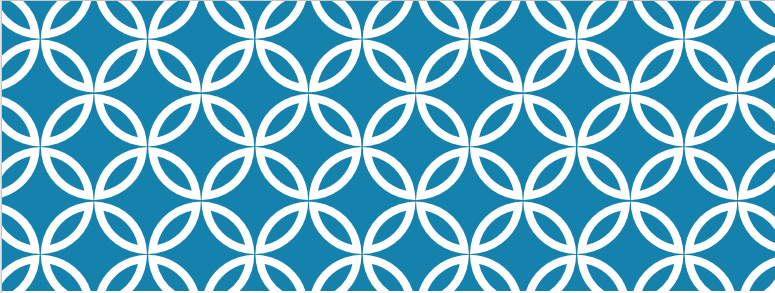
a



MOSIP

Modular Open Source Identity Platform

Version 0.1 | 12 Sep 2018

Packet Upload [FTP] - LLD

Copyright Information

This document is the exclusive property of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; the recipient agrees that they may not copy, transmit, use or disclose the confidential and proprietary information in this document by any means without the expressed and written consent of Mindtree. By accepting a copy, the recipient agrees to adhere to these conditions to the confidentiality of \_\_\_\_\_\_\_\_\_\_\_\_\_ practices and procedures; and to use these documents solely for responding to \_\_\_\_\_\_\_\_\_\_\_\_\_\_ operations methodology.

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver** | **Change Description** | **Sections** | **Date** | **Author** | **Reviewer** |
| 0.1 | First Draft | All | 12-Sep-18 | Sarvanan G | Omsai |

References

|  |  |  |  |
| --- | --- | --- | --- |
| No | Document Name | Ver. | Location |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Glossary

|  |  |  |
| --- | --- | --- |
| **Terminology** | **Definition** | **Remarks** |
| FTP | FILE TRANSFOR PROTOCAL |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[Copyright Information 2](#_Toc525842926)

[Revision History 2](#_Toc525842927)

[References 2](#_Toc525842928)

[Glossary 2](#_Toc525842929)

[Table of Contents 3](#_Toc525842930)

[Part A: Background 4](#_Toc525842931)

[1 Introduction 4](#_Toc525842932)

[1.1 Context 4](#_Toc525842933)

[1.2 Purpose of this document 4](#_Toc525842934)

[2 Scope 4](#_Toc525842935)

[2.1 Functional Scope 4](#_Toc525842936)

[2.2 Non Functional Scope 4](#_Toc525842937)

[3 Technical Approach 5](#_Toc525842938)

[3.1 Design Detail 5](#_Toc525842939)

[3.1.1 Validations: 7](#_Toc525842940)

[3.2 Class Diagram 7](#_Toc525842941)

[3.3 Sequence Diagram 7](#_Toc525842942)

[4 Success / Error Code 8](#_Toc525842943)

[5 Dependency Modules 9](#_Toc525842944)

[6 Database - Tables 9](#_Toc525842945)

[7 User Story References 9](#_Toc525842946)

Part A: Background

# Introduction

## Context

MOSIP is developed as an open source framework project. The java standard design principles will be followed to design the component.

## Purpose of this document

This document provides the low level technical design approach of a particular functionality in MOSIP Platform. It details out the in depth technical area of a particular scope.

# Scope

## Functional Scope

* Invoke the REST service to update the pre-sync status. [With multiple registration IDs which are under ‘yet to be synced’ status.]
* Once success then update the transaction and registration table.
  + Registration – client status with ‘S’
  + Transaction – update status with ‘Synched’
  + Timestamp update.
* Invoke the Upload REST service to push the list of packets to the server in a sequential manner [one by one]
* Once all packets pushed, update the respective status in the table.
  + Registration – client status with ‘P’
  + Transaction – update status with ‘Pushed’
  + Timestamp update.
* If there are any packets with the server status as ‘Resend’ then push that packet as well to the server and update the relevant status column.
* The role of the supervisor, which having the access to upload the packets should be able to browse the upload packet location.
* Able to provide the UI screen to the supervisor, where he can verify the upload information.
* The API should return the success / failure status code along with the respective message.

## Non Functional Scope

* Security :
  + The Enrollment packet shouldn’t be decrypt-able other than Enrollment Server.
  + FTP should be communicate via SSH private key always.
  + While uploading the packets folder, it should authenticate the user with username and password.
* Log the each state of the packet creation:
  + As a security measures the UIN or customer information should not be logged.
* Cache :
  + Enrollment packet data shouldn’t be cached and clear off all the data from the JVM local memory once the packet is created in local hard disk.
* Audit :
  + Each state of the packet upload should be stored into the DB for audit purpose.
  + UIN and important detail of the customer should not be audited.
* Exception :
  + Any exception occurred during the packet upload the same will be reported to the user with the user understandable exception.
* Data History :
  + The IDIS able to authenticate by using the Core Kernal module.
  + Maintain the Enrollment id, status and other high level info in the database table.
* Configuration:
  + SSH Private Key – the respective byte values will be present in the database table along with the expiry detail.
  + Before initiating the enrollment process, the key expiry to be validated.

# Technical Approach

## Design Detail

The detailed technical process for Uploading the Packet to the server is provided below:

**Packet Uploading:**

* Timestamp update.
* Invoke the Upload REST service to push the list of packets to the server in a sequential manner [one by one]
* Once all packets pushed , update the respective status in the table.
* Registration – client status with ‘P’
* Transaction – update status with ‘Pushed’
* Timestamp update.
* If there are any packets with the server status as ‘Resend’ then push that packet as well to the server and update the relevant status column.
* Create the **FileUploadController** with method **handleUpload** passing the ***filePath*** as a parameter.
* The component should get the uploading choose path and validate against the export path exists in the DB ***.[<Agency Code>/<Station Code>/<Date – Time Stamp>]***
  + ***Example: 2017[Agency Code] /72314[station code]/07-09-2018 18-24-33[DD-MM-YYYY HH-MM-SS]***
* Once the validation success, the component call the **AuthenticationController** to display the authenticate screen with username and password.
* Once the authentication got success it should redirect to FTPUploadController to upload the packets.
* **FTPUploadValidationService** having the method ***validate*** and the ***packetName*** is the parameter for the method to check the status before uploading to the enrolment server and to update the pre-sync status. [with multiple registration IDs which are under ‘yet to be synced’ status.]
  + Once success then update the transaction and registration table.
  + Registration – client status with ‘S’
  + Transaction – update status with ‘Synched’
* Create Java component API like “**FTPUploadManager**” and having the method name as “***uploadFile***“ and accepting the file as an argument to the method.
* Create the Java component like “***FTPConnectionService”*** as method as “connect” and the [url, sshkey, timeoutInterval, status] as a parameters to the method.
* Once all packets pushed , update the respective status in the table.
  + Registration – client status with ‘P’
  + Transaction – update status with ‘Pushed’
  + Timestamp update.
* If the folder already uploaded and only some packets are not uploaded the manager should check those packets and upload only those packets.
* Once the sure connection established the application able to transfer the each packet to the enrolment server and after successful upload it should update the status as UPLOADED.
* “***ENROLLMENT***” table (“***clientstatuscode***” column) as “Uploaded”.
* **“ENRL\_TRANSACTIONS**” table (insert the history and transaction data)
* The system should display the alert messages for success and failure messages.
* Once the uploaded is done, the API should be able to display the result of the upload as a UI screen having the below table with columns.
* If there are any packets with the server status as ‘Resend’ then push that packet as well to the server and update the relevant status column.

|  |  |
| --- | --- |
| Date Time | The uploaded Date time to server |
| Export Folder Name | The folder name of the exported folder |
| Uploaded | The count of the packets how many successfully uploaded |
| Not Uploaded/Rejected | The count of the packets how many not uploaded or rejected. |
| List of file names | We can display Map of the each packet name with status |
| Comments | Any other comments [Error s ….] |

**Assumptions:**

* The supervisor has the privileges to upload the packets.
* The packet status should be synched.
* The packet export should be happen from the same application.
* The packet is properly validated and approved.
* The export location where the packets resides the folder of [<Agency Code>/<Station Code>/<Date Time Stamp Folder>]
* Valid SSH key should be available for connect to the Enrollment server.

### Validations:

* Uploading file path should be validated against the DB path of the export folder path. [File Path should be : <folderpath>/<AgencyCode>/<Station Code>/<Date –Time Stamp>/<Each Packet ZIP>
* Always only one packet should be uploaded via FTP.
* Each packet status should be inserted to the ENROLLMENT and ENRL\_TRANSACTIONS table.
* User Authentication needs to be done.
* FTP connection status needs to be checked with SSH Key.
* Able to upload manually and automatic [batch job]
* While uploading only packets which are not uploaded should be upload.

## Class Diagram

[**https://github.com/mosip/mosip/blob/DEV/design/registration/\_images/\_class\_diagram/registration-packetupload-classDiagram.png**](https://github.com/mosip/mosip/blob/DEV/design/registration/_images/_class_diagram/registration-packetupload-classDiagram.png)

## Sequence Diagram

[**https://github.com/mosip/mosip/blob/DEV/design/registration/\_images/\_sequence\_diagram/registration-packetupload-sequenceDiagram.png**](https://github.com/mosip/mosip/blob/DEV/design/registration/_images/_sequence_diagram/registration-packetupload-sequenceDiagram.png)

# Success / Error Code

While uploading the packet we need to check the status of the packet upload from the server.

|  |  |  |
| --- | --- | --- |
| **Code** | **Type** | **Message** |
| REG-PCC-000 | Success | Status of the successful file transfer. |
| REG-PCC-001 | Error | Requested File action not taken. |

**Audit LOG:** Following status should be logged into the Audit Manager while processing the packets**.**

**DB Packet Table [Status code and description]:**

|  |  |
| --- | --- |
| **Code [Status\_Code]** | **Description** |
| C | Packet Encrypted and successfully created. |
| U | Packet Uploaded Successfully |
| S | Packet Meta information synched to server |
| D | Packet Deleted |
| A | Packet approved |
| R | Packet Rejected |
| H | Packet Hold on particular stage |
| E | Packet errors[ Ex : Virus scanner error] |
|  |  |

# Dependency Modules

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Module Name** | **Description** |
| FTP Uploader | Kernel | To upload the packets to the enrollment server |

# Database - Tables

* ENROLMENT
* ENRL\_TRANSACTIONS

# User Story References

|  |  |
| --- | --- |
| **User Story No.** | **Reference Link** |
| **MOS-559** | https://mosipid.atlassian.net/browse/MOS-559 |