*OEN Update Tool*

Technical Specification

Version 0.1

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| Nov 23, 2015 | 0.1 | Initial Draft | Tarique Ansari |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Approval

The purpose of the Technical Specification is to accomplish the functionality through architecture, integrations, and code. The Approvers listed below are agreeing on all aspects of design, system architecture, development, and implementation as defined in this document. Sign off is provided by the technical team only. This document will provide the technical solution as to how the Business Requirements will be met.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Signature** | **Approval Date** |
| Carlo Carlo Ammendolia | Internal Auditor |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Document Purpose 5](#_Toc436146285)

[1. Assumptions 5](#_Toc436146286)

[2. Infrastructure Diagrams 5](#_Toc436146287)

[2.1 Physical 5](#_Toc436146288)

[2.2 Application Access Mechanism 5](#_Toc436146289)

[3. Technical Specifications for Functional Specifications 7](#_Toc436146290)

[3.1 UI Component 9](#_Toc436146291)

[3.2 Server side Component 9](#_Toc436146292)

[4. Glossary/References 10](#_Toc436146293)

[5. Questions/Notes 10](#_Toc436146294)

[6. Appendix 10](#_Toc436146295)

Technical Specification

# Document Purpose

The purpose of the Technical Specification (TS) is to provide the system instructions that correlate to the Functional Requirements, Design, Data Mapping or Definitions, Architecture, or Logic to meet the Requirements requested by functional user.

# Assumptions

This tool assumes that the users are savvy business users with access to PFIS files and have great deal of confidentiality and security concerns.

# Infrastructure Diagrams

## Physical

This section is already documented on the Humber Wiki web content under the following links:

* <https://mywiki.humber.ca/display/ESPMAIN/Technology+Infrastructure>
* <https://mywiki.humber.ca/display/ESPMAIN/DEVL+-+Security+and+Access>
* <https://mywiki.humber.ca/display/ESPMAIN/Environments>

## Application Access Mechanism

Figure 1 is the current logical architectural depicting the flow of execution.



Figure

# Technical Specifications for Functional Specifications

As per Figure 2, the only actor involved is;

1. User

A User can only be savvy business user who has access to PFIS files.

User can access the application by hitting the below url in browser:

**http://NWL-Dev-Server-03.humber.org:3000/oenator/**

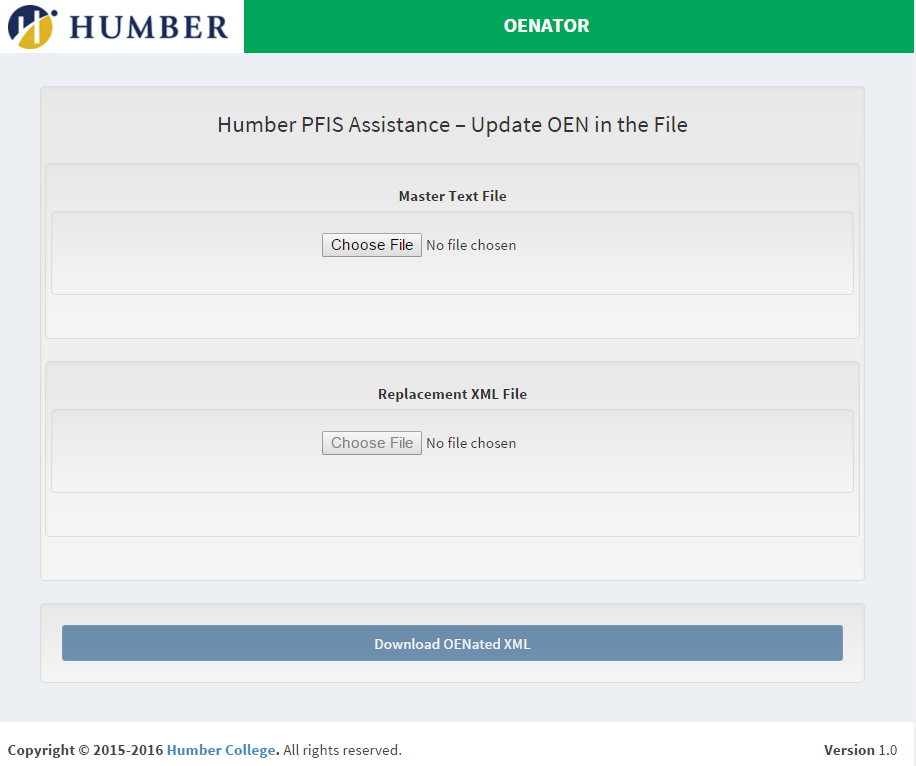


Figure 2

The abstracted higher level set of components is as follows.

## UI Component

User is presented with a nice looking UI built in Angular JS and bootstrap. Fields are disabled according to the use cases to avoid producing lot of error messages hence restricting user to make mistakes. User can select Master Text File consisting of the OEN number replaced with TOEN. Unless user selects the master text file OEN XML chooser remains disabled. Download button remains disabled until both master text file and OEN XML files are both chosen. As soon as the OEN XML file is chosen the processing starts in server side. Once the updated OEN is created Download button gets enabled allowing user to download the updated OEN file. After download is complete the download button get s disabled again for security.

## Server side Component

There are three main activities that happen in server side.

1. The master text file is parsed according to the regular expression which extracts the Spriden and its corresponding OEN from the file. The master text file usually has 50 thousands of lines to be streamed and parsed. Node JS is used to improve the streaming performance.
2. Once the Spriden and corresponding OENs are extracted, a JSON ocean of object is created with twenties of thousands of Spriden and OEN pairs. The JSON is kept in memory instead of being written in a file.
3. The last step involves parsing XML file and searching for OEN against each Spriden ID in the XML file from the JSON object. If an OEN is found corresponding to the Spriden ID , the OEN value for its Spriden ID is updated in the XML file. Once OEN update is complete it creates an updated XML and sends a signal to client. The client then enables the download button. Once the updated XML file is downloaded that very is deleted from server file system. All these happen synchronously in Node,.js
4. **Key Method Activities**
5. processtextfile

This method reads the Master Text File and created the JSON Ocean of object.

1. readupdatexml

This parses the XML file and picks each Spriden from the file.

1. findoen

This method searches the OEN for each Spriden from the JSON ocean of object in most efficient way possible.

1. writexmlfile

This writes the file in server file system with Spriden OEN.

1. downloadOenFile

This method sends the oenated xml file to the client and subsequently deletes the file from server file system.

|  |  |
| --- | --- |
| Requirement Traceability | |
| RQ-01  RQ-02  RQ-03 |  |

# Glossary/References

|  |  |
| --- | --- |
| TERM | DEFINITION |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Questions/Notes

# Appendix