HML and FHIR Conversion

Operation Manual

Andrew S. Brown, Ph.d. <abrown3@nmdp.org>

# Preface

We have developed a computational infrastructure to handle converting high volumes of raw Histoimmunogenetics Markup Language (HML) documents to a hierarchical collection of Fast Health Interoperability Resources (FHIR) based resources. This infrastructure is an ecosystem built around open source principals and currently implemented in Java and Python. It is theoretically possible to implement this project in any modern-day object-oriented language or framework; however, for the purpose of this implementation, the following prerequisites must be installed on a bash-capable operating system. To aid in the adoption of this implementation, we are currently writing build scripts that will execute in a Windows environment through PowerShell. Please check for release updates regularly as the development of this source code is an active effort, we strongly recommend keeping staying up to date with current code releases.

## Prerequisites:

* Maven v3.x+ (<https://maven.apache.org)>
* Python v2.7x+ (<https://www.python.org)>
* Java 8 (<http://www.oracle.com)>

# Environment Setup

This conversion infrastructure is designed to run in an environment consisting of a high-throughput message bus system, a map-reduce capable schemaless database, and a series of RESTful Application Programming Interfaces (APIs). We have chosen the following open-sourced projects to satisfy environmental requirements:

* Apache Kafka: high-throughput message bus (<https://kafka.apache.org)>
* MongoDB: schemaless map-reduce capable database (<https://www.mongodb.org)>
* Spring Framework: RESTful API framework (<http://projects.spring.io/spring-boot)>

## Kafka Setup:

1. Download Kafka v0.10.2.1 (<https://kafka.apache.org/downloads)>
2. Unzip, and copy the contents (3 files) of ‘kafka-setup’ directory included in this repository to ‘kafka\_2.12-0.10.2.1/config’ located in the kafka distribution downloaded in step 1
3. Start a single instance of zookeeper: ‘bin/zookeeper-server-start.sh config/zookeeper.properties’
4. Start a cluster of Kafka message brokers: ‘bin/kafka-server-start.sh config/server.properties & bin/kafka-server-start.sh config/server-1.properties & bin/kafka-server-start.sh config/server-2.properties’
5. This project depends on the creation of 3 topics: 1) hml-fhir-conversion 2) fhir-hml-conversion 3) fhir-submission
6. We strongly recommend the following replication schema for the creation of topics in step 5: 3 partitions, 3 replicates
7. Kafka Tool is an excellent GUI based administrative tool for managing your Kafka cluster, we recommend the creation of topics by this mechanism rather than the command line based approach (<http://www.kafkatool.com)>

## MongoDB Setup:

1. Download MongoDB v3.4.6 binary distribution (<https://www.mongodb.com/download-center?jmp=nav#community)>
2. Extract files from download in step 1
3. Copy files to desired target directory
4. Create a directory to house Mongo databases (recommend: /Database/Mongo)
5. Execute ‘mongodb –-dbpath={PATH\_TO\_DIRECTORY\_IN\_STEP\_4}’ from the target directory where binaries were copied
6. Execute the script (mongodb-setup/create\_databases.js) included in this release
7. We recommend the using MongoChef (https://studio3t.com) as a GUI to administer and manage your mongo database

# Building and Running RESTful APIs

Currently, there are 3 RESTful APIs that require building, installing, and running to support the implementation of this conversion system. The procedure to build, install, and run each of these APIs is identical and can be executed independently of one another to support ongoing development efforts, or built in a batch. To execute a batch build and install, simply download the repository marked ‘\*parent’ below.

## API Repositories:

* service-hml-fhir-converter-parent (\*parent) (<https://github.com/nmdp-bioinformatics/service-hml-fhir-converter-parent.git)>
* service-hml-fhir-converter-api (<https://github.com/nmdp-bioinformatics/service-hml-fhir-converter-api.git)>
* service-hml-fhir-converter (<https://github.com/nmdp-bioinformatics/service-hml-fhir-converter.git)>
* service-fhir-submission (<https://github.com/nmdp-bioinformatics/service-fhir-submission.git)>

## “Parent”-Install Process:

1. From service-hml-fhir-converter-parent directory, execute: ‘python Setup.py [options]’
2. Options are not required for building, ‘python Setup.py’ is adequate to begin process
3. Please see documentation for detailed explanation of optional build parameters (<https://github.com/nmdp-bioinformatics/service-hml-fhir-converter-parent)>

## Individual Module Install Process:

1. From directory of module source, execute: ‘sh install.sh’
2. To only execute build action, execute: ‘sh build.sh’

After installing, the executable Java Archive (JAR, .jar) (/target/[repository-name-version.jar] file can be run simply by executing ‘java –jar [PATH\_TO\_JAR]’. Optionally, the ‘-r’ flag can be passed, telling the install scripts to begin running the API server after install successfully completes.

# Building and Running Graphical User Interface (GUI) Tools:

### web-hml-fhir-converter

In an effort to both aid in the correct input of HML and FHIR data structures, we have created a GUI tool that currently supports the inputting of data through web forms. This data has machine-learning capability for auto-completion of structured data contained in various properties of an HML document. As the database grows in size and diversity, predictions to commonly populated discreet values can be obtained by entering the first few characters of its value. This future goal of this tool is support both the structured input of FHIR and HML as an entry point to creating mock data for use in the conversion ecosystem.

### web-hml-fhir-conversion-dashboard

In an effort to guide non-developer users of this conversion ecosystem, we have created a GUI tool that is capable of submitting files containing HML or FHIR data for conversion. This allows for the submission of FHIR or HML data into a conversion queue and tracks its status. Status results are returned as successful, processing, or error. Should an error be encountered, specific data on what caused the exception to fail the message is recorded and will be available to the user. After the conversion process has successfully completed, a file is available to download as either an HML document in XML or JSON format or FHIR a structured FHIR object that contains relational data for subsequent submission to a FHIR conformance server (XML or JSON). It is important to note at the time of writing this document, the FHIR conformance submission is an ongoing development effort and has not reached release status.

## Prerequisites:

* Node package manager (https://docs.npmjs.com/getting-started/installing-node)
* Bower package manager (install by: ‘npm install –g bower’)
* Ruby (<https://www.ruby-lang.org/en/downloads)>

## Repositories:

* web-hml-fhir-converter (<https://github.com/nmdp-bioinformatics/web-hml-fhir-converter.git)>
* web-hml-fhir-conversion-dashboard (<https://github.com/nmdp-bioinformatics/web-hml-fhir-conversion-dashboard.git)>

## Building and Deploying (from root directory of repository):

1. ‘gem install compass’
2. ‘npm install –g grunt grunt-cli’
3. ‘npm install’
4. ‘bower-install’
5. ‘grunt-serve’

# Building and Running Kafka Consumer Processes

Currently, there are 3 Kafka consumer APIs that require building, installing, and running to support the implementation of this conversion system. The procedure to build, install, and run each of these APIs is identical and can be executed independently of one another to support ongoing development efforts, or built in a batch. To execute a batch build and install, simply download the repository marked ‘\*parent’ below.

## Consumer Repositories:

* process-kafka-hml-fhir-parent (\*parent) (<https://github.com/nmdp-bioinformatics/process-kafka-hml-fhir-parent.git)>
* process-kafka-fhir-hml-conversion-consumer (<https://github.com/nmdp-bioinformatics/process-kafka-fhir-hml-conversion-consumer.git)>
* process-kafka-hml-fhir-conversion-consumer (<https://github.com/nmdp-bioinformatics/process-kafka-hml-fhir-conversion-consumer.git)>
* process-kafka-fhir-submission-consumer (<https://github.com/nmdp-bioinformatics/process-kafka-fhir-submission-consumer.git)>

## “Parent”-Install Process:

1. From service-hml-fhir-converter-parent directory, execute: ‘python Setup.py [options]’
2. Options are not required for building, ‘python Setup.py’ is adequate to begin process
3. Please see documentation for detailed explanation of optional build parameters (<https://github.com/nmdp-bioinformatics/service-hml-fhir-converter-parent)>

## Individual Module Install Process:

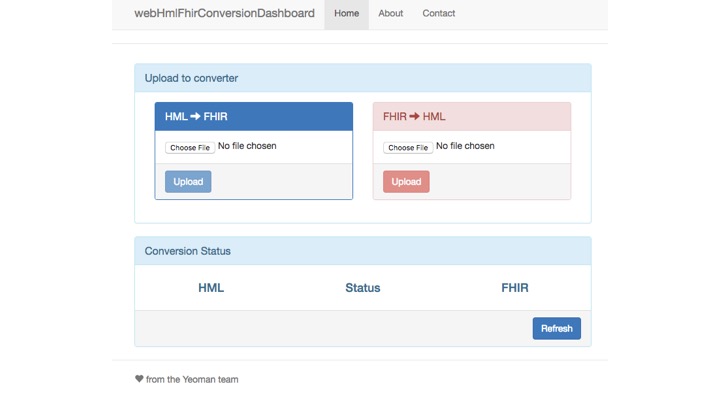
1. From directory of module source, execute: ‘sh install.sh’
2. To only execute build action, execute: ‘sh build.sh’

After installing, the executable Java Archive (JAR, .jar) (/target/[repository-name-version.jar] file can be run simply by executing ‘java –jar [PATH\_TO\_JAR]’. Optionally, the ‘-r’ flag can be passed, telling the install scripts to begin running the API server after install successfully completes.

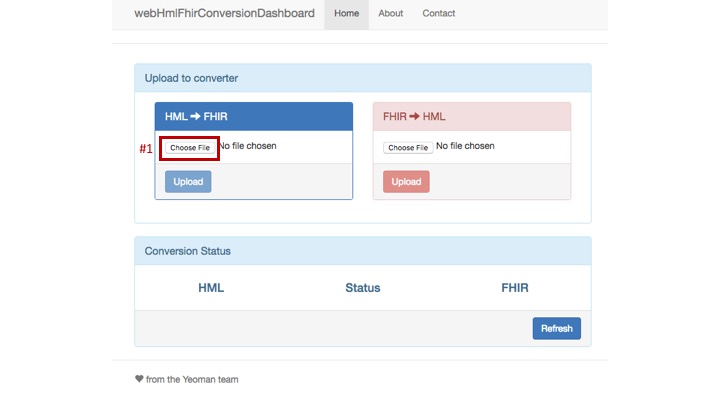
# Supplemental

### GUI Conversion Tool (web-hml-fhir-conversion-dashboard)

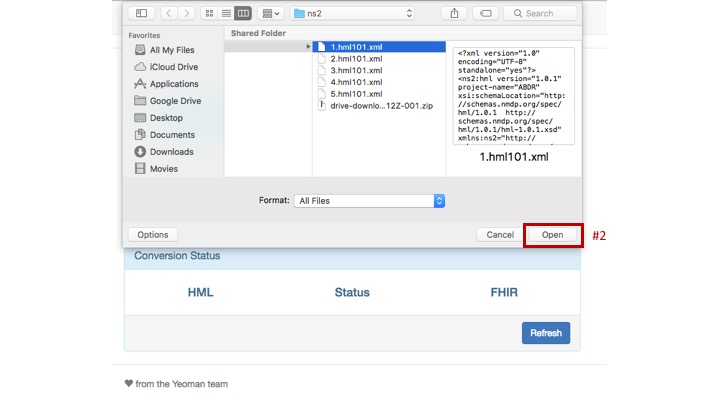
This is a single page application (SPA), you will be taken to the initial upload page



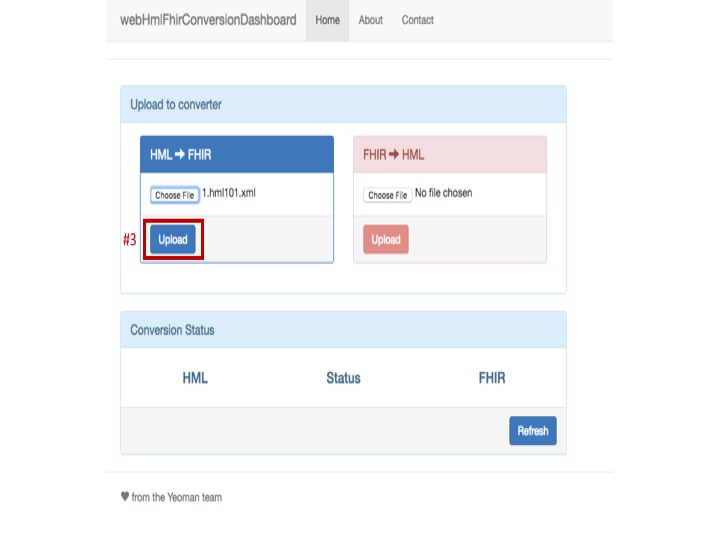
From the upload page, begin by uploading an HML file by clicking ‘Choose File’ button (#1)



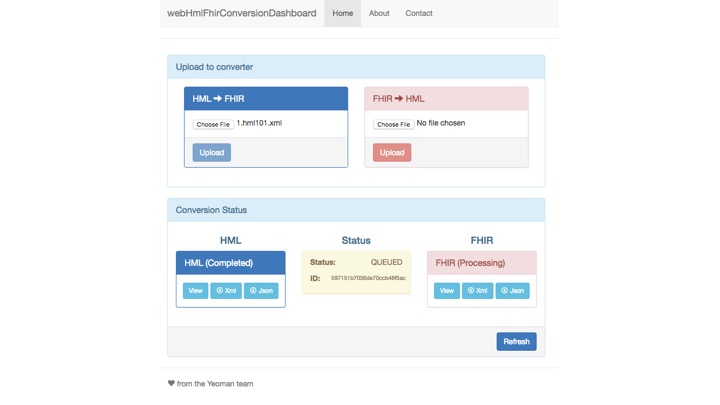
Select a valid file and click ‘Open’ button (#2)



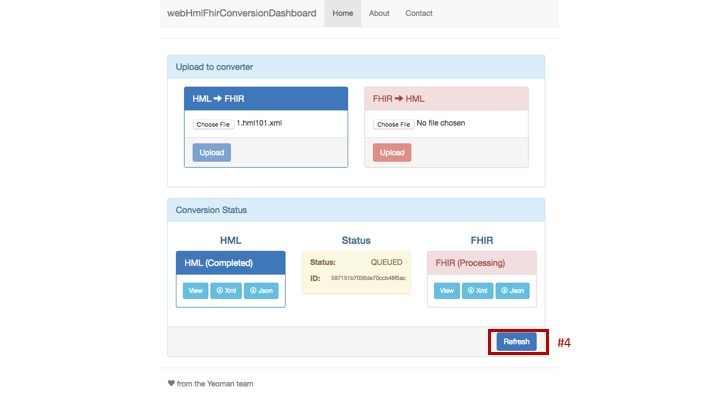
Upload file by clicking ‘Upload’ button (#3)

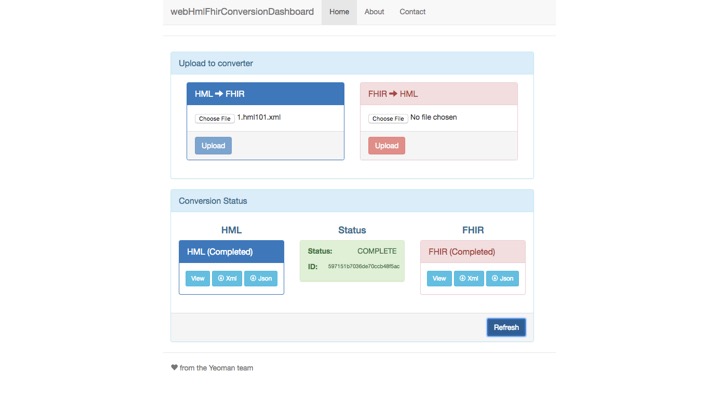


The page will automatically refresh and show the current status of the conversion process



Clicking the ‘Refresh’ (#4) button will update the status of the converted message





To download files as JSON, click in the desired layout’s ‘Json’ button (#6), to download as XML, click in the desired layout’s ‘Xml’ button (#7). An upcoming feature will allow visualization without the need of downloading files (‘View’ buttons).

