OpenFurther Reference Documentation

Table of Contents

1. About

The following documentation applies to **OpenFurther version 1.4.0-SNAPSHOT**

1.1. Conventions

A note

An important point

A tip

A warning

A point of caution

2. Introduction

OpenFurther is an informatics platform that supports federation and integration of data from heterogeneous and disparate data sources.

It has been deployed at the University of Utah (UU) as the Federated Utah Research and Translational Health e-Repository (FURTHeR) since August 2011 and is available for use by all U of U employees and students. OpenFurther links heterogeneous data types, including clinical, public health, biospecimen and patient-generated data; empowering researchers with the ability to assess feasibility of particular clinical research studies, export biomedical datasets for analysis, and create aggregate databases for comparative effectiveness research. With the ability to link unique individuals from these sources, OpenFurther is able to identify cohorts for clinical research.

It provides semantic and syntactic interoperability as it federates health information on-the-fly and in real-time and requires neither data extraction nor homogenization by data source partners, facilitating integration by retaining data in their native format and in their originating systems.

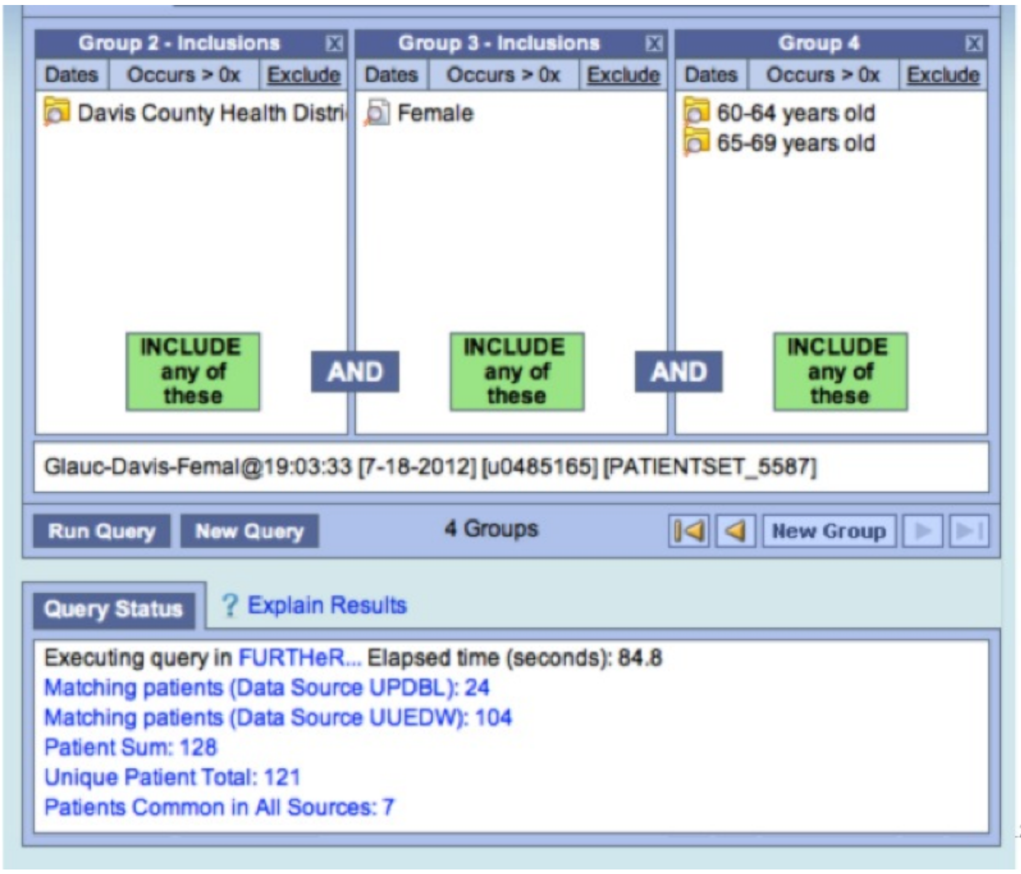
OpenFurther is built upon Maven, Spring, Hibernate, ServiceMix, and other open source frameworks that promote OpenFurther’s code reusability and interoperability.

3. Architecture

Loosely, OpenFurther runs as a multi-tier application. The presentation layer or front end/user-interface is served (currently) through the i2b2 web client. The logic layer is served through the ServiceMix ESB, and the database layer is served using Oracle 11g, although it can be configured for other databases as well.

3.1. User Interface

OpenFurther utilizes the i2b2 web client as a front-end for querying data. The user interface has been modified to support federated querying.



Glauc-Davis-Femal@19:03:33 [7-18-2012] [u0485165] [PATIENTSET_5587]

```
Executing query in FURTHeR... Elapsed time (seconds): 84.8
Matching patients (Data Source UPDBL): 24
Matching patients (Data Source UUEDW): 104
Patient Sum: 128
Unique Patient Total: 121
Patients Common in All Sources: 7
```

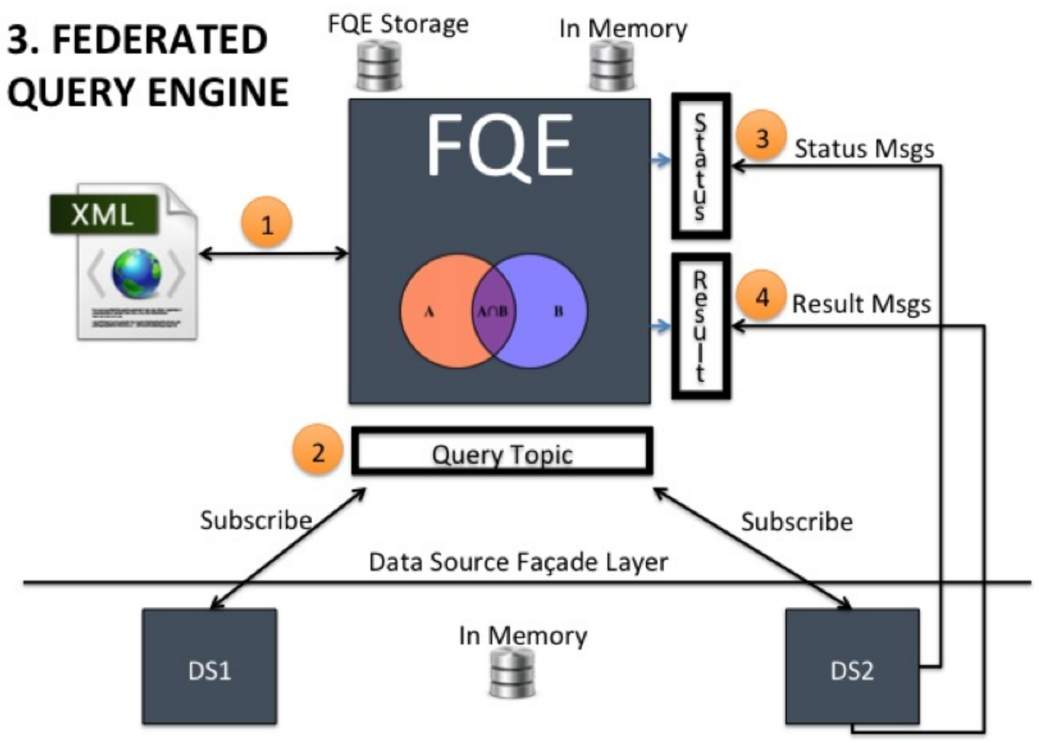
3.2. Hooking OpenFurther into i2b2

OpenFurther utilizes a Java Servlet Filter to divert query requests to the OpenFurther backend system. The Servlet filter looks for XML messages from i2b2 that indicate a query is being run. Those XML messages are then diverted to OpenFurther where OpenFurther converts them into a OpenFurther query and runs them. All other XML messages are ignored and i2b2 is allowed to run as normal.

No data is stored within i2b2, all data resides within its original location

3.3. The Federated Query Engine (FQE)

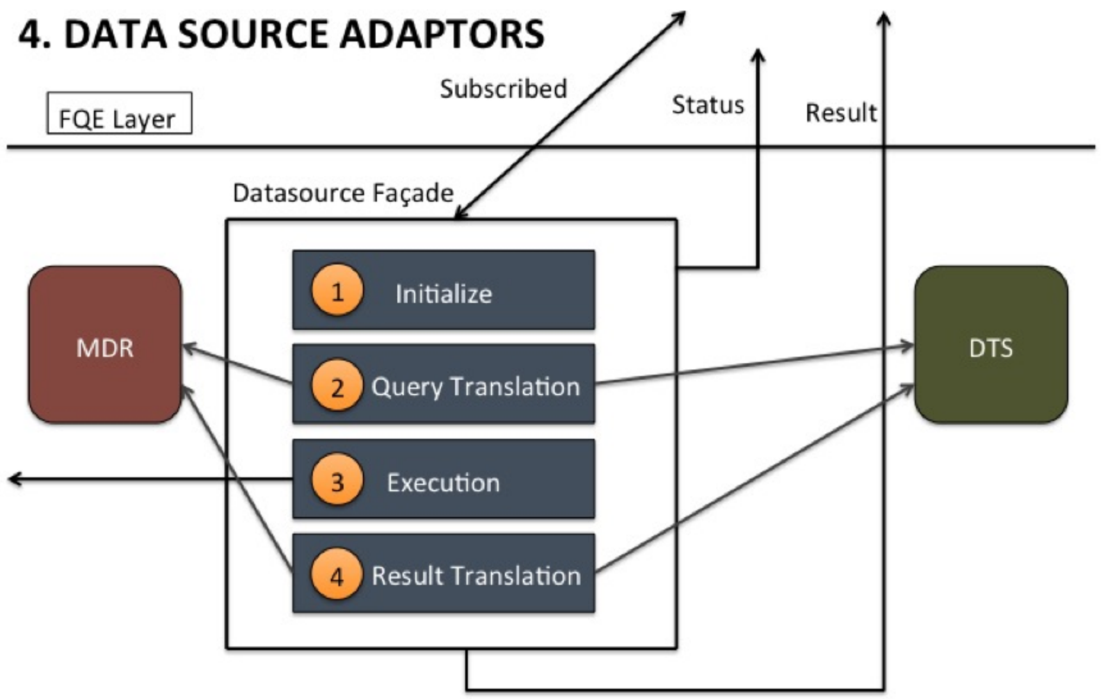
In OpenFurther, the term "FQE" (Federated Query Engine) is broadly referred to as the set of software modules involved in the execution of a federated query.



- A federated query written in FQL (an XML based query language) or an i2b2 query is submitted at **1**.
- Utilizing the publish-subscribe pattern, one or more data source adapters are subscribed to the Query Topic at **2**.
- If the query is an i2b2 query, the FQE converts the i2b2 query to a federated query.
- The FQE then posts the query to the Query Topic (**2**) and each listening data source adapter receives a copy of the query.
- Each data source adapter runs through a number of steps to initialize, process, and translate a query for a given data source (Explained below).
- Throughout the processing, status messages are sent to a Status Queue at **3**.
- Once results are translated to a common model, they are persisted to the In-Memory database and result count is sent to **4**.

3.4. Data Source Adapters

Data source adapters are facades around an existing data source. Data source adapters can be entirely custom for any given implementation or they can use a pre-written adapter if their data source is already in a well-known format such as OMOP, i2b2, OpenEMR, etc



- Data source adapters follow the chain-of-responsibility pattern. The process of adapting a query is broken down into several small steps and each output is passed on to the next step. Data source adapters typically have 4 commons steps.

  1. They are given an initialization step which allows them to determine whether or not the given data source can answer the given query. It also provides for any other initialization required throughout the process.

  2. Query translation translates the logical FQL that is not specific to any data source into data source specific language. This will vary with data sources. Some data sources will utilize SQL, other's might be a web service. It utilizes the Metadata Repository (MDR) for translating attributes and values (e.g. logical query uses Gender but actual data source uses Sex as the attribute). It also utilizes DTS (Terminology Server) to translate from a given code (e.g. ICD9 250) to the data source's code (e.g. 12345)

  3. The query is executed against the data source and results are returned in their native format (SQL ResultSet, XML, etc).

  4. Result translations translates the results into a common model with standardized vocabulary/ terminology utilizing the Metadata Repository (MDR) and DTS (Terminology Server).

3.5. Terminology Server

OpenFurther utilizes a terminology server (currently from Apelon DTS) to resolve the differences between coded values. This terminology server is responsible for storing the codes for standards as well as mappings between standards. It is also utilized to store data source specific codes (local codes) and mappings to standard namespaces (ICD9, SNOMED, CPT, etc)

Apelon DTS

The Apelon DTS (Distributed Terminology System) is an integrate set of open source components that provides comprehensive terminology services in distributed application environments.

DTS Supports national and international data standards, which are a necessary foundation for comparable and interoperable health information, as well as local vocabularies.

DTS consists of

- DTS Core - the core system, database, api, etc
- DTS Editor - a GUI interface for viewing, adding, and editing concepts
- Dts Browser - a web interface for viewing concepts
- Modular Classifier - allows for extending standard ontologies

3.6. The Metadata Repository (MDR)

The MDR is responsible for storing information (artifacts) about varying data sources. This includes things like data models, attributes, attribute types, etc. It is accessed using web services.

- Home grown but follows standards
  - XMI, Dublin Core
  - HL7 datatypes, CDA, DDI
- Stores artifacts
  - Logical models (UML), local models (UML), model mappings
  - Administrative information
  - Descriptive information
- Models supported
  - OMOP, i2b2, local models

4. Technologies

OpenFurther is built on a number of Open Source technologies

- Languages
  - Java
  - Groovy

  - Bash
  - Python
- Development Tools
  - Maven 3
  - SonaType Nexus
  - Eclipse
  - Git
  - JIRA
  - Bamboo
- Service Frameworks
  - Spring
  - Apache Commons
  - Apache CXF
  - Apache Camel
- Application Servers
  - Apache ServiceMix
- Testing
  - JUnit
  - Spock

5. Installing

OpenFurther is provided as a VM image for download at this time. The VM can be used as a reference for installation, typically splitting out each Linux user as an individual server.

TODO: Expand this section with detailed instructions for installing on Linux and Windows

6. Demo System Administration

OpenFurther utilizes a number of different servers to run. The following instructions pertain to the demo VM of OpenFurther that is available for download. All scripts used for starting and stopping services are available within the further-open-extras repository on GitHub.

The demo version contains all of the servers as individual Linux users.

6.1. Apache HTTP Server

The Apache HTTP server runs on port 80 and port 443. As root, run the following

```
service httpd start|stop
```

6.2. In-Memory Database Server

The HSQLDB server runs on port 9001. As root, run the following

```
/etc/init.d/hsqldb start|stop
```

6.3. Core Database Server

While our architecture supports different database, we’ve currently only tested OpenFurther on Oracle and Oracle XE

```
service oracle-xe start|stop
```

6.4. Terminology Server

The terminology server (Apelon DTS) runs on port 16666 (Requires that the Oracle Database Server has started). As root, run the following

```
su - dtsdemo
dts-auto start|stop
```

6.5. Enterprise Service Bus (ESB)

OpenFurther utilizes an ESB (Apache ServiceMix) to run application code. The ESB requires that the in-memory database, core database, and terminology server are already started. As root, run the following

```
su - esb
start_esb
```

To stop the ESB:

```
su - esb
esbl
further@localhost’s password:
further@local> shutdown
Confirm: shutdown instance local (yes/no):
```

6.6. Logging Locations

Apache HTTP Server

The Apache HTTP server logs are located in /var/www/httpd/

In-Memory Database Server

The HSQLDB is currently not configured for logging

Core Database Server

The Oracle XE database server is currently not configured for logging

Terminology Server

The Apelon DTS server logs in /home/demodts/Apelon_DTS/dts/bin/logs

Enterprise Service Bus (ESB)

ServiceMix ESB logs in /home/esb/servicemix/data/log

OpenFurther-i2b2

FURTHeR-i2b2 logs in 2 different locations * jboss: /home/i2b2/jboss/server/default/logs * tomcat: /home/i2b2/tomcat/logs