Centric Consulting CCD Merge Architecture and Technical Specifications

# Introduction

The Centric Consulting CCD Merge application is designed to overcome the burdens of manual merging of HL7 Continuity of Care Documents (CCD) for individual patients so that they can be easily indexed and securely stored within health information exchanges. The CCD Merge application is designed to make the merge process easy to initiate for any business person, not just IT staff, as well, as easing the process of defining merge rules so that non-IT staff are able to modify existing merge rules and add new merge rules.

# Technical Overview

The CCD Merge application is built on several modules which are depicted in the following diagram.



1. The user interface consists of 2 main views: a file submitter view which allows files to be submitted manually for merge, and an audit viewer which allows the results of merge attempts to be viewed. The audit view will also allow both the merged file itself, as well as the source documents for the merged document to be downloaded.
2. A web/rest service façade is provided as an API for the CCD Merge application. The User Interface accesses this API on behalf of an end user. In addition, this layer will have a directory watcher component which will observe an FTP directory and submit files from this directory for merge operations from an automated perspective.
3. The file merger component merges the documents together. It is divided into 3 sub-components.
   1. The input parser parses the xml content of submitted documents into an object model that will be further used by the code.
   2. The rules processor reads configured business rules from the rules repository and applies applicable business rules to the merge operations of the document. The rules processor is responsible for submitting audit information.
   3. The output builder is responsible for taking the elements that have been determined to be part of the merge result and writing them to an output file.
4. The rules engine consists of a set of natural language business rules that state the business rules for merging CCDs into a single CCD.
5. The audit log consists of the source documents as well as the merged document, who submitted the document for merge, the date and time the merge took place, and the business rules utilized to facilitate the merge.
6. The Data Store is currently configured to utilize the PostgresSql database management system. The data store contains the audit log tables and the pre-merged and merged documents in a compressed format.

# Minimum Requirements

## Application Server

* Windows, Mac, or Linux machine
* Java SDK version 6 or higher
* Play Framework version 2.1.3 available from <http://www.playframework.com/download>
* There is a standalone web server provided with the Play Framework, or utilize external webservers such as Apache, lighttpd, nginx etc. Web server configuration directions can be found here: <http://www.playframework.com/documentation/2.2.x/HTTPServer>

## Database Server

* Windows 2000 SP4 or later, Mac OS X, most recent Linux distributions, various Unix installations (full list available at <http://www.postgresql.org/docs/9.1/static/supported-platforms.html>)
* PostgresSQL version 9.1 available at (<http://www.postgresql.org/download/>)

# Deployment

While it is technically possible to deploy all of the CCD Merge application to a single server running both the web application and the database server this is not recommended for several reasons in a production environment. It is recommended to install the database on a separate server to insure additional security as well as allowing for further scalability of the application. Overall performance will also be improved by having processing distributed over more CPUs.

It is recommended to only expose the CCD Merge application through SSL and Secure FTP. In addition, if possible set up IP filtering rules on the web server to block traffic from unknown addresses and/or subnets. The next several diagrams will depict possible deployment configurations.

## Distributed Deployment



This configuration provides the most scalability, as well as higher security. The application is exposed to the end user via a Load Balancer/Proxy server that sits inside of a DMZ behind a firewall that has holes punched in it for an Https port, and possibly an FTPS port if an FTP directory is utilized for dropping files. The Load Balancer would then have access to one or more web application servers that would sit inside of another firewall. The web servers themselves would be accessible from the Load Balancer IP address. The load balancer would be configured to evenly distribute network traffic between the web servers. Data would be stored on one or more Postgres Database Servers in a database cluster.

## Simple Deployment



This is a simple deployment. The web/application server sits directly behind a firewall with an Https port exposed, and possibly a Ftps port if ftp file transfers are allowed to be made directly into an ftp directory. The PostgresSql database runs on a single separate database server.

## Cloud Deployment



The CCD Merge application can be easily deployed to Cloud Envrionments such as Heroku. Heroku has support for both the Playframework and Heroku. Deployment instructions for Heroku can be found at the following url: <http://www.playframework.com/documentation/2.2.x/ProductionHeroku>.

# Data Model



These tables store the audit information.

# Installation:

1. Obtain the CCD Merge source code.
2. Create the ccd\_merge database within your PostgresSql instance
3. Create a user specific for the web server to run under such as “wwwrun” and grant the appropriate permissions to the “ccd\_merge” database. These will need to be configured in your application.conf file. (The application.conf file can be found in the following directory: /conf/application.conf) The configuration file should contain database settings similar to the following:

db.default.driver=org.postgresql.Driver

db.default.url="jdbc:postgresql://localhost/ccd\_merge"

db.default.port=5432

db.default.user=wwwrun

db.default.password=wwwrun

1. Run the database\_schema.sql script on your databases to install the database tables. This script can be found in the source code under the following directory: /data/database\_schema.sql
2. Build the code for distribution with the following command
   1. \* Nix systems

$ play dist

* 1. Windows systems

C:\your source directory> play dist

1. Deploy the zip file to your application servers.
   1. Special note for \*Nix systems

For Unix users, zip files do not retain Unix file permissions so when the file is expanded the start script will be required to be set as an executable:

$ chmod +x /path/to/bin/<project-name>

Alternatively a tar.gz file can be produced instead. Tar files retain permissions. Invoke theuniversal:package-zip-tarball task instead of the dist task:

play universal:package-zip-tarball

1. Navigate to the target directory on the web server and type in:
   1. \*Nix systems

$ play clean stage

* 1. Windows systems

C:\your install directory> play clean stage

# Scaling the Application

Scaling of the application is dependent on the hardware it is running on, as well as the number of concurrent transactions and documents being processed at a given time. It is also dependent on the network backbone the application is running on. With that being said, scaling of the application should be extremely simple. The Play Framework, which is at the core of this application, is stateless and allows for easy deployment to multiple application servers. A load balancer can then distribute the load to the various application servers.

In addition, the back-end database, PostgreSQL, allows itself to be deployed into a clustered database environment. At the current time the documents are not stored in an external datasource, but could be applied if the volume within the database gets too big to store the data on the primary database server.

Since the application is currently in a beta release no recommendations can be provided as to when would be an appropriate time to scale the application with the above techniques. In addition, current resources are not available to setup a true performance testing environment that would assist in determining these points.

# Skills Required to Enhance and Support the Application

The following skillsets are required to support this application:

1. At least basic Sql skills, although knowledge of PostgreSql would be helpful.
2. Basic knowledge to run applications from command line interfaces, but prior experience with the Play Framework is ideal.
3. Basic knowledge of Scal and the Java Software Development Kit verison 6 or higher
4. To add business rules knowledge of the Drools framework is required.