SAMHSA HealthIT | HEALTH INFORMATION TECHNOLOGY

Behavioral Health Information Technology and Standards (BHITS) Project

Consent2Share Version 2 Deployment Guide

October 2016

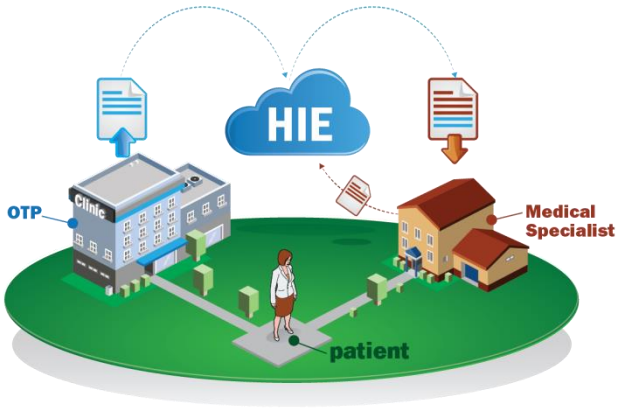
Prepared by FEi Systems

Contents

1 Introduction

1.1 Overview

Specially protected health information (PHI) covered under the federal confidentially regulation 42 CFR Part 2 (health information from federally assisted drug and alcohol treatment programs) has generally not been included in the electronic exchange of patient information between health care providers. One of the primary reasons is the lack of technology options for patients to share part of their health information while not sharing others.



To address this issue, the Federal Office of the National Coordinator (ONC) for Health Information Technology developed the Data Segmentation for Privacy (DS4P) initiative to allow patients to share portions of an electronic medical record while not sharing others. In collaboration with ONC, the Substance Abuse and Mental Health Services Administration (SAMHSA) developed Consent2Share to address the specific privacy protections for substance use treatment patients covered by 42 CFR Part 2.

Consent2Share is an open source tool for consent management and data segmentation. It is designed to integrate with existing Electronic Health Record (EHR) and Health Information Exchange (HIE) systems. SAMHSA funded the Open Behavioral Health Information Technology Architecture (OBHITA), to develop the Consent2Share application. SAMHSA funded the Behavioral Health Information Technology and Standards (BHITS) project to further develop and conduct pilot testing of Consent2Share. Through a process of electronic consent, the patient controls how their sensitive health data will be shared by selecting categories to segment in clinical documents such as C32, CCD, and other document types specified in C-CDA.

1.2 Purpose

This document was prepared by the application developers of the Consent2Share application primarily to document key infrastructure setup, installation, configuration, and deployment technologies required to operationalize Consent2Share. This document is not a step-by-step software application development guide. Rather, this document is a reference guide to help developers and system administrators install, configure, and deploy the key software components that operationalize Consent2Share. Since Consent2Share is designed to integrate with existing medical record system via interoperability standards, this document assumes that the reader has in place various HIE components and is employing interoperability standards. However Consent2Share application can be configured to work without HIE connection. Consent2Share development team has HIEOS and OpenEMPI installed as the HIE environment, refer to the Appendix 4.1 for more information.

1.3 Organization of this Guide

This Deployment Guide is divided into four Chapters:

- ❑ Chapter One provides an introduction to Consent2Share and the purpose of this guide.
- ❑ Chapter Two provides information about setting up the deployment environment based on Docker.
- ❑ Chapter Three provides instructions to deploy and configure Consent2Share using Docker on Linux servers.
- ❑ Chapter Four provides appendix

1.4 Technology Stack

The current version of Consent2Share (Consent2Share Version 2) employs Docker. In short, Docker is an open-source tool that automates the deployment of applications inside software containers. Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run. This includes code, runtime, system tools, and system libraries. This enables the ability that the application will always run the same, regardless of the environment within it is running.

Consent2Share is designed to integrate with existing HIE systems via interoperability standards. To do so, the technology stack for Consent2Share configuration is as follows. The references for these technologies located in the Appendix 4.2.

- Docker 1.12.2 and Docker Compose 1.8.1
- Apache Tomcat Version 8
- Drools Guvnor Version 5.5.0
- Logback Audit Server Version 0.6.1
- ClamAV 0.98.7
- MySQL Version 5.7
- HIEOS Version 1.2
- OpenEMPI Version 2.2.9

1.5 Prerequisites

This document is designed for developers or system administrators who install, configure, deploy, and maintain distributed applications. Familiarity with the following is recommended.

- Basic Linux system administration
- Basic knowledge of Docker and Docker-Compose
- Basic knowledge of Public Key Infrastructure (PKI) and creating SSL certificates

1.6 Technical Support

If you have specific questions about a specific API deployment, setup server environment, or anything related to the Consent2Share application, you should:

- Check the Consent2Share Project site
- Check the readme files available for each API in Consent2Share GitHub repository.
- Check the Issues in consent2share repository.

2 Deployment Server Set Up

2.1 Docker Installation

The following provides instructions about how to install Docker on a Linux CentOS 7.X server.

2.1.1 Prerequisites

- Docker requires a 64-bit installation regardless of your CentOS version.
- Your kernel must be 3.10 at a minimum, which CentOS 7 runs. To check the CentOS version, run the command "`uname -r`" in the terminal.
- User account should have sudo or root privileges
- Ensure yum and cURL are installed, and networking is operational.

2.1.2 Install Docker

- Log into the CentOS 7 server with sudo or root privilege user.
- Ensure that existing YUM packages are up to date.

```
sudo yum update
```

- Run the Docker installation script.

```
sudo curl -fsSL https://get.docker.com/ | sh
```

- Enable the Docker service to run automatically after reboot.

```
sudo systemctl enable docker.service
```

- Start the Docker daemon/engine.

```
sudo systemctl start docker
```

- Verify the Docker installation.
  - `sudo systemctl status docker`
    Should show the active (running) status
  - `sudo docker run hello-world`

  Output message will contain the following:

  Hello from Docker!

  This message shows that your installation appears to be working correctly.

2.1.3 Install Docker Compose

Run the following two commands to install Docker Compose:

- Install Docker Compose:

```
sudo curl -L https://github.com/docker/compose/releases/download/1.8.1/docker-compose-
`uname -s`-`uname -m` > /usr/local/bin/docker-compose
```

- Apply executable permissions to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

- Verify Docker compose install by checking its version:

```
sudo docker-compose --version
```

Note: if docker-compose gives the command “not found” try with following:

```
/usr/local/bin/docker-compose --version
```

2.1.4 Add User Accounts to Docker Group

The user accounts that need to run Docker and Docker Compose commands must be added to the Docker group. Run the following command by replacing the *** with the actual username to add a user to the Docker group

```
sudo usermod -aG docker ***
```

3 Consent2Share Deployment

Two server deployment options are provided to run Consent2Share application on Linux. Here we use CentOS 7.X as an example to describe the setups.

Consent2Share Docker images will be downloaded from Dockerhub BHITS public registry.

3.1 One Server Setup

This option is designed to run all Consent2Share services, UIs and databases on a single server.

3.1.1 Configure

- Create a new directory ‘/usr/local/java/C2S_PROPS’

```
mkdir /usr/local/java/
mkdir /usr/local/java/C2S_PROPS
```

- If SELinux is enabled, run the command below to assign the relevant SELinux policy type as a workaround to prevent issues while mounting volumes to the containers from ‘/usr/local/java’

```
chcon -Rt svirt_sandbox_file_t /usr/local/java
```

- Get the c2s_docker.sh file and place it in the ‘/etc/profile.d/’ folder
  - Modify the C2S_APP_HOST and SMTP variables according to the server environment
  - Re-login to the server in order for the file `c2s_docker.sh` to run automatically during the login
  - Verify by checking any variable mentioned in the file

    Example: echo ${C2S_BASE_PATH} should show the value set in the file

- Create the following sub folders under ‘/usr/local/java/C2S_PROPS’ folder:
  - uaa
  - pls-api/config-template
  - pls-api/init-db
  - logback-audit/config-template
  - logback-audit/init-db
  - iexhub/temp
  - iexhub/test
- Copy the following configuration files and other files, place them under the ‘/usr/local/java/C2S_PROPS’:

- Copy the uaa.yml to uaa sub folder
- Copy all pls-api config-temlate files to pls-api /config-template sub folder
- Copy the sample provider data sql file to pls-api/init-db sub folder
- Copy the logback-audit_config-template file to logback-audit/config-template sub folder
- Copy the database schema sql file to 'logback-audit/init-db' sub folder
- Copy all IExHub resource folder files into iexhub/temp sub folder

➢ Get the docker-compose file and place it in the '/usr/local/java' folder

➢ Modify the following configuration files
- In /usr/local/java/C2S_PROPS/pls-api/config-template/pls-config.properties file, replace 'localhost' with 'pls-db.c2s.com' in the variable **database.url**
- In /usr/local/java/C2S_PROPS/uaa/uaa.yml file, replace 'localhost' with 'uaa-db.c2s.com' in the variable **database.url**

➢ Edge-server security:
- Create a new directory named 'keystore' under '/usr/local/java' folder
- Create/Obtain a valid SSL certificate
- Export the public and private keys from the SSL certificate to a JKS formatted keystore file named 'edge-server.keystore'
- Put the 'edge-server.keystore' file into '/usr/local/java/keystore' folder
- Modify the value of the 'server.ssl.key-store-password' property in the 'docker-compose.yml' file located in the '/usr/local/java' folder to match the password used when exporting/creating the SSL certificate

3.1.2 Compose Containers

Run the following command from the '/usr/local/java' folder to start up all Consent2Share services, UIs and databases:

```
docker-compose up -d
```

Run 'docker ps –a' to verify all the containers are up running except data-only containers.

3.2 Two Servers Setup

This option is to run Consent2Share services, UIs on an application server and databases on a separated database server.

3.2.1 Configure Database Server

➢ Create a new directory '/usr/local/java/C2S_PROPS' folder

```
mkdir /usr/local/java

mkdir /usr/local/java/C2S_PROPS
```

➢ If SELinux is enabled, run the command below to assign the relevant SELinux policy type as a workaround to prevent issues while mounting volumes to the containers from '/usr/local/java'

```
chcon -Rt svirt_sandbox_file_t /usr/local/java
```

➢ Get the c2s_docker.sh file under '/etc/profile.d/' folder

- Uncomment the C2S_DB_HOST variable
- Modify the C2S_APP_HOST, C2S_DB_HOST and SMTP variables according to the server environment.
- Re-login to the server in order for the file c2s_docker.sh to run automatically during the login
- Verify by checking any variables mentioned in the file
  Ex: echo ${C2S_BASE_PATH} should show the value set in the file

➢ Create the following sub folders under '/usr/local/java/C2S_PROPS'

- pls-api/init-db
- logback-audit/init-db

➢ Get the following files under the /usr/local/java/C2S_PROPS.

- Get the sample provider data sql file into pls-api/init-db sub folder
- Get the database schema sql file into logback-audit/init-db sub folder

➢ Get the docker-compose-db-server file as docker-compose.yml under the `/usr/local/java` folder

3.2.2 Configure Application Server

➢ Create a new directory '/usr/local/java/C2S_PROPS'

```
mkdir /usr/local/java/
mkdir /usr/local/java/C2S_PROPS
```

➢ If SELinux is enabled, run the command below to assign the relevant SELinux policy type as a workaround to prevent issues while mounting volumes to the containers from '/usr/local/java'

```
chcon -Rt svirt_sandbox_file_t /usr/local/java
```

➢ Get c2s_docker.sh file under '/etc/profile.d/' sub folder

- Uncomment the C2S_DB_HOST variable
- Modify the C2S_APP_HOST, C2S_DB_HOST and SMTP variables according to the server environment
- Re-login to the server in order for the file `c2s_docker.sh` to run automatically during the login
- Verify by checking any variable mentioned in the file
  Ex: echo ${C2S_BASE_PATH} should show the value set in the file

➢ Create the following sub folders under '/usr/local/java/C2S_PROPS' folder

- uaa
- pls-api/config-template
- logback-audit/config-template
- iexhub/temp
- iexhub/test

➢ Copy the following configuration files and place them under the `/usr/local/java/C2S_PROPS`

- Copy the uaa.yml to uaa sub folder
- Copy all pls-api config-temlate files to pls-api /config-template sub folder
- Copy the logback-audit config-template file to logback-audit/config-template sub folder
- Copy all IExHub resource folder files to iexhub/temp folder

➢ Get the docker-compose-app-server file as docker-compose.yml under the '/usr/local/java' folder

➢ Modify the following configuration files
- In /usr/local/java/C2S_PROPS/pls-api/config-template/pls-config.properties file, replace 'localhost' with the value of *C2S_DB_HOST* in the variable **database.url**
- In /usr/local/java/C2S_PROPS/uaa/uaa.yml file, replace 'localhost' with the value of *C2S_DB_HOST* in the variable **database.url**

➢ Edge-server security:
- Create a new directory named 'keystore' under '/usr/local/java' folder
- Create/Obtain a valid SSL certificate
- Export the public and private keys from the SSL certificate to a JKS formatted keystore file named 'edge-server.keystore'
- Put the 'edge-server.keystore' file into '/usr/local/java/keystore' folder
- Modify the value of the 'server.ssl.key-store-password' property in the 'docker-compose.yml' file located in the '/usr/local/java' folder to match the password used when exporting/creating the SSL certificate

3.2.3 Compose Containers on Database Server

Run the following command from the '/usr/local/java' folder to start up all databases:

```
docker-compose up -d
```

```
Run 'docker ps –a' to verify all the containers are up running except data-only containers.
```

3.2.4 Compose Containers on Application Server

Run the following command from the '/usr/local/java' folder to start up all Consent2Share services, UIs:

```
docker-compose up -d
```

```
Run 'docker ps –a' to verify all the containers are up running.
```

3.3 Resolve Deployment Error

If you encounter an error in the deployment:

ERROR: for dss.c2s.com UnixHTTPConnectionPool(host='localhost', port=None): Read timed out. (read timeout=60)

Follow the steps below to resolve the error:

1. Restart the Docker service: `sudo service docker restart`
2. Check for all Docker containers that are running: `docker ps -a`
   If you notice any containers that are exited or down except the data-only containers based on `busybox` image, follow the next steps
3. For instance, if MySql containers are not running
   a. Go to /usr/local/java and then remove all containers : `docker-compose down`
   b. Go to /usr/local/java and then remove mysql folder : `sudo rm -rf mysql/`

4. Start up all containers: Re-run from '/usr/local/java' folder: `docker-compose up -d`

3.4 UI Urls

- Consent2Share Admin UI: https://<application_server>/admin-ui
  - By default, Consent2Share comes with a provider staff admin user
  - Login to Consent2Share Admin UI as an admin using username 'consent2share@gmail.com ' and password 'admin'
  - Follow the Consent2Share Admin User Guide to verify Consent2Share admin features
- Consent2Share Patient UI: https://< application_server >/pp-ui

  - Follow the Consent2Share Patient User Guide to verify Consent2Share patient features
    - By default, Consent2Share comes with ten providers. Adding providers to create consents may be done using the following information.
    - Sample clinical documents can be used to verify 'Upload Medical Documents' and 'Try My Consent Settings against My Medical Record before sharing' features.

3.5 Generate and Reconfigure UAA Public and Private Keys

By Default Consent2Share has created public and private keys that are available in the configuration files. It's recommended to change them especially in production environment.

- Create a temporary folder uaa-keystore under /usr/local/java

- Go to uaa-keystore folder, Run following in command line to generate a pair of public and private key. Enter pass phrase when promoted.
  - `openssl genrsa -des3 -out uaa_token_key_private.pem 2048`
  - `openssl rsa -in uaa_token_key_private.pem -outform PEM -pubout -out uaa_token_key_public.pem`
  - `openssl rsa -in uaa_token_key_private.pem -out uaa_token_key_private_unencrypted.pem -outform PEM`

- Update uaa.yml under /usr/local/java/C2S_PROPS/uaa.
  - Replace jwt.token.verification-key with public key in uaa_token_key_public.pem.
  - Replace jwt.token.signing-key with private key in `uaa_token_key_private_unencrypted.pem`.

- Update c2s_docker.sh under /etc/profile.d
  - Replace UAA_PUBLIC_KEY with public key in uaa_token_key_public.pem.
    - Ensure no spaces in the key value, check the default value for reference
  - Re-login to the server for latest UAA_PUBLIC_KEY to be effective.

4 Appendix

4.1 HIE Environment Setup

Consent2Share development team has HIEOS and OpenEMPI installed as HIE in development environment to work with Consent2Share application.

- HIEOS Version 1.2

HIEOS is an open source implementation of, primarily server-side, Integrating the Healthcare Enterprise (IHE) Integration Profiles including Cross Enterprise Document Sharing (XDS.b) and Cross Community Access (XCA) integration profiles.

Please follow the links at below to download HIEOS and install it on your server. Once it is installed, you can configure Consent2Share application to work with HIEOS by following the steps below:

- Disable the test mode in Information Exchange Hub (IExHub) by setting the `TestMode` property to `false` in `/usr/local/java/C2S_PROPS/iexhub/temp/IExHub.properties` file.
- Set the XDS.b configurations accordingly in `/usr/local/java/C2S_PROPS/iexhub/temp/IExHub.properties` file. The related configurations are:
  - XdsBRegistryEndpointURI
  - LogXdsBRequestMessages
  - XdsBRepositoryEndpointURI
  - XdsBRepositoryUniqueId
  - XdsBKeyStoreFile
  - XdsBKeyStorePwd
  - XdsBHttpsProtocols
  - XdsBCipherSuites
  - XdsBSubmissionSetOid
  - XdsBDocumentClassCodesNodeRepresentation
  - XdsBDocumentClassCodesNodeRepresentationContract
  - XdsBDocumentClassCodesCodingScheme
  - XdsBDocumentClassCodesName
  - XdsBDocumentFormatCodesNodeRepresentation
  - XdsBDocumentFormatCodesCodingScheme
  - XdsBDocumentFormatCodesName
  - XdsBDocumentHealthcareFacilityTypeCodesNodeRepresentation
  - XdsBDocumentHealthcareFacilityTypeCodesCodingScheme
  - XdsBDocumentHealthcareFacilityTypeCodesName
  - XdsBDocumentPracticeSettingCodesNodeRepresentation
  - XdsBDocumentPracticeSettingCodesCodingScheme
  - XdsBDocumentPracticeSettingCodesDisplayName
  - XdsBSubmissionSetUniqueOidPrefix

Reference: http://sourceforge.net/projects/hieos/

TLS Configuration: https://kenai.com/projects/hieos/pages/SetupGuide#TLS_Configuration

Applications: http://sourceforge.net/apps/mediawiki/hieos/index.php?title=HIEOS_1.2_-_Architecture

- OpenEMPI Version 2.2.9

Open Enterprise Master Patient Index is an open source implementation of an Enterprise Master Patient (EMPI) which is a repository that maintains a registry of all patients across an enterprise. Please follow the links at below to download OpenEMPI and install it on your server. Once it is installed, you can configure Consent2Share application to work with OpenEMPI by following the steps below:

- Enable HIE connection in `patient-registration` service by setting `HIE_CONNECTION_ENABLE` value in `c2s_docker.sh` to `true`
- Disable the test mode in Information Exchange Hub (IExHub) by setting the `TestMode` property to `false` in `/usr/local/java/C2S_PROPS/iexhub/temp/IExHub.properties` file.
- Set the PIX and PDQ manager configurations accordingly in `/usr/local/java/C2S_PROPS/iexhub/temp/IExHub.properties` file. The related configurations are:
  - PIX:
    - PIXManagerEndpointURI
    - LogPIXRequestMessages
    - LogPIXResponseMessages
    - PIXKeyStoreFile
    - PIXKeyStorePwd
    - PIXHttpsProtocols
    - PIXCipherSuites
    - PIXReceiverApplicationName
    - PIXReceiverApplicationRepresentedOrganization
    - PIXProviderOrganizationName
    - PIXProviderOrganizationContactTelecom
    - PIXProviderOrganizationOID
    - PIXQueryIdOID
    - PIXDataSourceOID
  - PDQ:
    - PDQManagerEndpointURI
    - LogPDQRequestMessages
    - LogPDQResponseMessages
    - PDQKeyStoreFile
    - PDQKeyStorePwd
    - PDQHttpsProtocols
    - PDQCipherSuites
    - PDQReceiverApplicationName
    - PDQReceiverTelecomValue
    - PDQQueryIdOID
    - PDQOtherIDsScopingOrganizationOID
    - PDQReceiverApplicationRepresentedOrganization

Reference: http://www.openempi.org/

Installation:

https://kenai.com/projects/openempi/downloads?field=date&order=desc

http://www.openempi.org/confluence/display/openempi227/Installation+Instructions

Test Tools: http://www.openempi.org/confluence/pages/viewpage.action?pageId=7995511

4.2 Technology References

- Docker: 1.12.2 and Docker-Compose: 1.18.1

Reference: https://www.docker.com
Installation Files: https://docs.docker.com/engine/installation/linux/centos/

- Apache Tomcat Version 8

Reference: http://tomcat.apache.org/
Installation Files: http://tomcat.apache.org/download-80.cgi

- MySQL Version 5.7

Reference: http://www.mysql.com/
Installation Files: http://www.mysql.com/downloads/

- Drools Guvnor Version 5.5.0

Reference: http://drools.jboss.org/drools-guvnor
Installation Files: docker image: `docker pull bhits/guvnor:5.5.0`

- Logback- Audit Server Version 0.6.1

Reference: http://audit.qos.ch/index.html
Installation Files: docker image: `docker pull bhits/logback-audit-server:0.6.1`

- ClamAV:0.98.7

Reference: http://www.clamav.net/