SAMHSA HealthIT
HEALTH INFORMATION TECHNOLOGY

Behavioral Health Information Technology and Standards (BHITS) Project

Consent2Share Version 3.2.0 Deployment Guide

June 2017

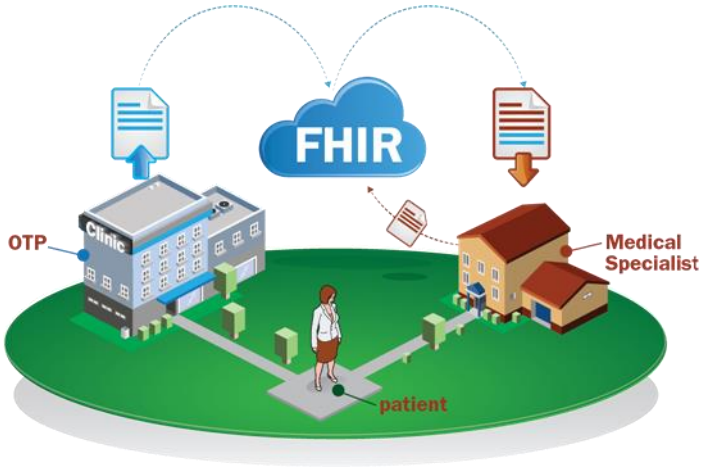
Prepared by FEi Systems

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 for Health Information Technology (ONC) 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 the ONC, the Substance Abuse and Mental Health Services Administration (SAMHSA) developed the Consent2Share application to address the specific privacy protections for substance use treatment patients covered by the federal confidentiality regulation 42 CFR Part 2.

Consent2Share is an open source application for data segmentation and consent management. It is designed to integrate with existing FHIR (Fast Health Interoperability Resources) systems. Initially, SAMHSA funded the Open Behavioral Health Information Technology Architecture (OBHITA) contract to develop the Consent2Share application. Subsequently, SAMHSA funded the Behavioral Health Information Technology and Standards (BHITS) contract to further develop and conduct pilot testing of Consent2Share. Through a process of electronic consent, the patient controls how his or her sensitive health data will be shared by selecting categories.

1.2 Purpose

This document was prepared by the application developers of Consent2Share 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, deploy, and validate the key software components that operationalize Consent2Share. Since Consent2Share is designed to integrate with FHIR (Fast Health Interoperability Resources) Systems, this document assumes that the reader has in place FHIR Servers and is employing interoperability standards. However, the Consent2Share application can be configured to work without FHIR connections. The Consent2Share development team has an instance of HAPI FHIR Restful Server installed as the FHIR environment. Please 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 is an appendix that provides information about FHIR Server setup

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 FHIR 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.

- AngularJS
- Angular Material
- Angular CLI
- Node.js
- npm
- MD2
- RxJS
- TypeScript
- JavaScript - ES6
- HTML5
- CSS3
- Oracle Java 8
- Spring Framework
- Spring Boot
- Spring Cloud
- Apache Maven
- Apache Tomcat
- MySQL
- Flyway
- Docker and Docker Compose
- Cloud Foundry User Account and Authentication (UAA) Server

1.5 Prerequisites

This document is designed for developers and 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 Setup

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 and Docker Compose

- Get the c2s_docker_install.sh and run the file.

  sh c2s_docker_install.sh

- Verify the Docker installation.

  sudo docker version

  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.

- Verify Docker compose installation.

  sudo docker-compose --version

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

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

2.1.3 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 the 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 Server Info

	MAX	MIN
Memory	65 GB	20GB
Storage	80GB	25GB
CPU	2	1

3.1.2 Configure

- Get the c2s_config.sh and run the file.

  curl https://raw.githubusercontent.com/bhits-dev/consent2share/master/infrastructure/scripts/c2s_config.sh > c2s_config.sh

  sh c2s_config.sh oneServerConfig

  Expected Results:

  The following subfolders and the Consent2Share configurations will be created under '/usr/local/' folder.

  - Java
    - ✓ docker-compose file : docker-compose.yml
  - Application Files
    - java /C2S_PROPS/uaa
      - ✓ uaa configuration file: /uaa/uaa.yml
    - java /C2S_PROPS/c2s-config-data
      - ✓ Consent2share configuration files: /c2s-config-data
    - java /keystore
    - The c2s_one_server_env file will be placed as **c2s_env.sh** file in '/etc/profile.d/' folder.
  - Database Files
    - java /C2S_PROPS/pcm
      - ✓ insert_consent_attestation_term.sql file:

- /pcm/ insert_consent_attestation_term.sql

✓ insert_consent_revocation_term.sql file:

- /pcm/ insert_consent_revocation_term.sql

✓ insert_purposes.sql file:

- /pcm/ insert_purposes.sql

- java /C2S_PROPS/phr
  - ✓ Document Type file: /phr/insert_document_type_codes.sql
- java /C2S_PROPS/pls
  - ✓ Pls sample provider data file: /pls/pls_db_sample.sql
- java /C2S_PROPS/vss
  - ✓ VSS sample provider data file: /vss/vss_db_sample.sql
- java /C2S_PROPS/ums
  - ✓ Administrative Gender Code file: /ums/ insert_administrative_gender_code_lookup_data.sql
  - ✓ Country Code file : /ums/ insert_country_code_lookup_data.sql
  - ✓ Locale file: /ums/ insert_locale_lookup_data.sql
  - ✓ Identifier file: /ums/insert_npi_identifier_system.sql
  - ✓ Role sample file: /ums/insert_role_scopes_lookup_data.sql
  - ✓ State Code file : /ums/insert_state_code_lookup_data.sql

➢ Edge-server security:

- 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'
- upload the 'edge-server.keystore' file into '/usr/local/java/keystore' folder
- Add the following in the '/usr/local/java/C2S_PROPS/c2s-config-data/edge-server.yml' file.

```
spring.profiles: your_app_Server_specific_profile

server:

  ssl:

    key-store: /java/keystore/edge-server.keystore

    key-store-password: "keystore password"
```

- Modify the ENV_APP_PROFILE= your_app_Server_specific_profile in oneServerConfig() function in the '/etc/profile.d/ c2s_env.sh' file.

➢ Modify the following configuration files

- Set edge server, config server, and SMTP variables to the server specific values in the '/etc/profile.d/ c2s_env.sh' file.
- Replace 'localhost' with 'uaa-db.c2s.com' for **database.url** variable in '/usr/local/java/C2S_PROPS/uaa/uaa.yml' file.
- There are many Consent2Share API codes. The configurations in these codes can be overridden using the corresponding API YAMLs that are available in the **c2s-config-**

data folder. The structure of the API YAMLS should be similar to the corresponding application.yml mentioned in the each API code. They can be found in the 'src/main/resources' folder.

- ✓ For instance, to override database variables for PCM API. The following can be added in the pcm.yml as below
  spring.profiles: your_app_Server_specific_profile
  spring:
  datasource:
  url: "database url"
  username: "database user name"
  password: "{cipher} encypted database pwd"
- ✓ Follow the instructions mentioned in the config server api to encrypt/decrypt server specific sensitive configurations.

➢ Re-login to the server in order for the file ` c2s_env.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

3.1.3 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 and UIs on an application server and databases on a separate database server.

3.2.1 Server Info

APP Server	MAX	MIN
Memory	40 GB	20GB
Storage	80GB	15GB
CPU	2	1

DB Server	MAX	MIN
Memory	20GB	10GB
Storage	80GB	10GB
CPU	2	1

3.2.2 Configure Database Server

➢ Get the c2s_config.sh and run the file.

curl https://raw.githubusercontent.com/bhits-dev/consent2share/master/infrastructure/scripts/c2s_config.sh > c2s_config.sh

sh c2s_config.sh twoServerDbConfig

Expected Results:

The following subfolders and the Consent2Share configurations will be created under '/usr/local/' folder.

- Java
  - ✓ docker- compose-db-server file : docker-compose.yml
  - java /C2S_PROPS/pcm
    - ✓ insert_consent_attestation_term.sql file:
      - /pcm/ insert_consent_attestation_term.sql
    - ✓ insert_consent_revocation_term.sql file:
      - /pcm/ insert_consent_revocation_term.sql
    - ✓ insert_purposes.sql file:
      - /pcm/ insert_purposes.sql
  - java /C2S_PROPS/phr
    - ✓ Document Type file: /phr/insert_document_type_codes.sql
  - java /C2S_PROPS/pls
    - ✓ Pls sample provider data file: /pls/pls_db_sample.sql
  - java /C2S_PROPS/vss
    - ✓ VSS sample provider data file: /vss/vss_db_sample.sql
  - java /C2S_PROPS/ums
    - ✓ Administrative Gender Code file: /ums/ insert_administrative_gender_code_lookup_data.sql
    - ✓ Country Code file : /ums/ insert_country_code_lookup_data.sql
    - ✓ Locale file: /ums/ insert_locale_lookup_data.sql
    - ✓ Identifier file: /ums/insert_npi_identifier_system.sql
    - ✓ Role sample file: /ums/insert_role_scopes_lookup_data.sql
    - ✓ State Code file : /ums/insert_state_code_lookup_data.sql
  - The c2s_two_servers_db_env file will be placed as c2s_env.sh file in '/etc/profile.d/' folder.

➢ Re-login to the server in order for the file c2s_env.sh file 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

3.2.3 Configure Application Server

➢ Get the c2s_config.sh and run the file.

curl https://raw.githubusercontent.com/bhits-dev/consent2share/master/infrastructure/scripts/c2s_config.sh > c2s_config.sh

sh c2s_config.sh twoServerAppConfig

Expected Results:

The following subfolders and the Consent2Share configurations will be created under '/usr/local/' folder.

- Java
  - ✓ docker-compose-app-server file : docker-compose.yml
- Application Files
  - java /C2S_PROPS/uaa
    - ✓ uaa configuration file: /uaa/uaa.yml
  - java /C2S_PROPS/c2s-config-data
    - ✓ Consent2share configuration files: /c2s-config-data
  - java /keystore
  - The c2s_two_servers_app_env.sh file will be placed as c2s_env.sh file in '/etc/profile.d/' folder.

➢ Edge-server security:

- 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'
- upload the 'edge-server.keystore' file into '/usr/local/java/keystore' folder
- Add the following in the '/usr/local/java/C2S_PROPS/c2s-config-data/edge-server.yml' file.

```
spring.profiles: your_app_Server_specific_profile

server:

 ssl:

  key-store: /java/keystore/edge-server.keystore

  key-store-password: "keystore password"
```

- Modify the ENV_APP_PROFILE= your_app_Server_specific_profile in twoServerAppConfig() function in the '/etc/profile.d/ c2s_env.sh' file.

➢ Modify the following configuration files

- Set edge server, config server, and SMTP variables to the server specific values in the '/etc/profile.d/ c2s_env.sh' file.
- There are many Consent2Share API codes. The configurations in these codes can be overridden using the corresponding API YAMLs that are available in the c2s-config-data folder. The structure of the API YAMLS should be similar to the corresponding application.yml mentioned in the each API code. They can be found in the 'src/main/resources' folder.
  - ✓ For instance, to override database variables for PCM API. The following can be added in the pcm.yml as below

```
spring.profiles: your_app_Server_specific_profile
spring:
 datasource:
  url: "database url"
  username: "database user name"
  password: "{cipher} encypted database pwd"
```

- ✓ Follow the instructions mentioned in the config server api to encrypt/decrypt server specific sensitive configurations.

➢ Re-login to the server in order for the file ` c2s_env.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

3.2.4 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.

3.2.5 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 Populate sample data

3.3.1 Sample Providers (pls)

➢ Login to docker pls database container

```
docker ps  | grep pls-db
docker exec -it  <<pls-db container id>> bash
```

➢ Run pls_db_sample.sql from container directory ` /java/C2S_PROPS/pls`.

```
cd  /java/C2S_PROPS/pls
mysql -p
    enter root pwd(default is admin)
source pls_db_sample.sql
```

3.3.2 Sample Value Sets (vss)

➢ Login to docker vss database container

```
docker ps  | grep vss-db
docker exec -it  <<vss-db container id>> bash
```

➢ Run pls_db_sample.sql from container directory ` /java/C2S_PROPS/vss`.

```
cd  /java/C2S_PROPS/vss
mysql -p
    enter root pwd(default is admin)
source vss_db_sample.sql
```

3.3.3 Sample Data for Consent Management (pcm)

- Login to docker pcm database container

  docker ps | grep pcm-db

  docker exec -it <<pcm-db container id>> bash

- Run pcm sample files from container directory ` /java/C2S_PROPS/pcm`.

  cd /java/C2S_PROPS/pcm

  mysql -p

  enter root pwd(default is admin)

  source insert_consent_attestation_term.sql

  source insert_consent_revocation_term.sql

  source insert_purposes.sql

3.3.4 Sample Data for User Management Service

- Login to docker ums database container

  docker ps | grep ums-db

  docker exec -it <<ums-db container id>> bash

- Run pcm sample files from container directory ` /java/C2S_PROPS/ums`.

  cd /java/C2S_PROPS/ums

  mysql -p

  enter root pwd(default is admin)

  source insert_administrative_gender_code_lookup_data.sql

  source insert_country_code_lookup_data.sql

  source insert_locale_lookup_data.sql

  source insert_role_scopes_lookup_data.sql

  source insert_state_code_lookup_data.sql

3.3.5 Sample Document Type Code (phr)

- Login to docker pls database container

  docker ps | grep phr-db

  docker exec -it <<phr-db container id>> bash

- Run phr_db_sample.sql from container directory ` /java/C2S_PROPS/phr`.

  cd /java/C2S_PROPS/phr

  mysql -p

enter root pwd(default is admin)

source insert_document_type_codes.sql

3.4 Possible Deployment Errors

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

- *ERROR: for any issues while mounting volumes to the containers from '/usr/local/java'*

  If SELinux is enabled, run the command below to assign the relevant SELinux policy type as a workaround to prevent mounting issues.

  sudo chcon -Rt svirt_sandbox_file_t /usr/local/java

3.5 UI URLs

- Consent2Share Staff UI: https://<application_server>/staff-ui
  - By default, Consent2Share comes with a staff admin user c2s-admin@mailinator.com.
  - Login to Consent2Share UI as a patient using username 'c2s-admin@mailinator.com' and password 'AAA#aaa1'
  - Follow the Consent2Share Patient User Guide to verify Consent2Share staff admin features

By default, Consent2Share comes with sample providers as documented in Section#3.3. Please refer to Consent2Share Patient User Guide to add providers and then create consents

- Consent2Share UI: https://<application_server>/c2s-ui
  - Follow the Consent2Share Patient User Guide to verify Consent2Share patient features
    - ✓ By default, Consent2Share comes with sample providers as documented in Section#3.3. Please refer to Consent2Share Patient User Guide to add providers and then create consents.

3.6 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 is 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 HAPI FHIR Server Installation

The Consent2Share version of the HAPI FHIR Server docker image is available at the Docker Hub. Please refer to the full description of the image and run it accordingly.