# OCP Architecture information

General Areas of Discussion

- The dozen or so different servers and their function, especially

ocp-discovery-server (Eureka), ocp-config-server,

- document-validator: validates C-CDA and C32 documents
- dss: Document Segmentation Service. It has APIs to segment, redact/tag C-CDA documents, FHIR resource and bundle.
- vss: Value Set Service. This contains value set categories, value sets and the concept codes that are associated with those value sets. DSS uses these mappings to figure out if a particular concept code found in a FHIR resource or C-CDA document is sensitive or not. We load the sensitive codes in VSS. DSS can add the value set category codes to FHIR resource security label metadata.
- ocp-uaa: As I mentioned in my previous email, it is a fork of CloudFoundry UAA, OAuth2 authorization and authentication server.
- ocp-hapi-fhir-jpaserver: An implementation of FHIR server based on HAPI FHIR API, it also includes the authorization/security for SMART on FHIR scopes. All medical related data in OCP is stored on FHIR server.
- ocp-fis: FHIR Integration Service. This is an integration service used by other services when connecting to FHIR server. Basically, the other services don't directly call FHIR, they go through ocp-fis.
- ocp-ui: Main user interface implemented using React as a single page application.
- ocp-ui-api: This is the API meant to serve ocp-ui, following Backend-for-Frontend pattern. Typically the services ending with -api are also responsible with security, they would have a security configuration securing the endpoints with OAuth2 scopes.
- smart-core: The core service to support SMART on FHIR endpoints. OCP uses UAA as the OAuth2 server, but for SMART on FHIR apps, they actually use these endpoints via the smart-gateway.
- smart-gateway: This is just a reverse proxy securing certain endpoints on smart-core.
- ocp-config-data: just a git repository that was meant to store external configuration files.
- ocp-config-server: As mentioned in my previous email, this is the service used by other microservices to load the configurations. For instance ocp-configs-server can be configured to load and serve configurations from ocp-config-data repo/folder.
- ocp-discovery-server: mentioned in my previous email
- ocp-edge-server: This is based on Spring Cloud Netflix Zuul. It is used as a reverse proxy (api gateway) and configured to expose certain services only.
- c2s-sof-ui: Single Page react app for the UI of Consent2Share SMART on FHIR app
- c2s-sof-api: Backend part of C2S SoF app. This is basically just utilized by ui for calling FHIR server, kind of similar purpose to ocp-fis.

Just a couple hints about services:

- see the *.web packages to see the REST controllers, which may give you an idea about the interface of that service
- check edge-server route configuration to see which services are actually exposed, the others can be considers as internal core services.

- What parts of ocp are completely open source vs. what parts are proprietary (such as Quippe, Consent2Share)

--- only the OCP framework with care coordination and C2S are open source. Everything else that was launched is proprietary. One of the benefits of the Smart on FHIR (SoF) launch capability is that any SoF available application can be used in the OCP. Typically, it is way cheaper and better to use applications that are already built, than to try and build them from scratch.

- Patient Profile creation - confirm OCP supports Patient profile creation

- I believe you should be able to create patients if you login with a user with Organization Admin role.

- Reporting features of OCP? ---- There are no reporting features currently in OCP.

- The overall deployment process for Consent 2 Share.

- You can technically deploy C2S SoF app anywhere as long as it is accessible to the user's browser and it can access the exposed smart-endpoints and FHIR server. Imagine this is like a 3rd party app that is being launched from the platform that could be running on somewhere else.
- In server environment, we were using docker and docker-compose to run all services.

Other Questions:

1) What are the minimum number of OCP servers modules to have a successful user login to the OCP UI and use most functions? Here's what we know for now -

ocp-ui, uaa-api, uaa-db. Is ocp-ui-api (and therefore ocp-config-server and ocp-discovery-server required)

- In development environment, I think you would need minimum discovery-server, config-server, ocp-ui, ocp-ui-api, ocp-uaa, ocp-fis, FHIR server just to login. However, I recommend to run all services to make sure something is not left behind. edge-server may not be needed if you open up the UI on the nodejs development server with `npm start` localhost:3000

2) Where can we find error logging for the OCP UI and API's?

- check loggin.file configurations in application.yml. these can be reconfigured as well.

3) What is involved with deployment of Consent 2 Share? For example we know a separate UAA is required, but what else is necessary to integrate with (OCP)?

- as mentioned above and earlier, just for C2S SoF app, 2 projects only. But, you need a SoF platform to launch it, so the entire OCP infrastructure needs to be up, and then you need to register the C2S SoF app on OCP. Later, you can launch that as an app on patient profile.

OCP admin role can register SoF apps for the entire platform.

A few roles including care coordinator can launch SoF apps from patient's context:

4) What facilities for data export and import does OCP offer?

----- Currently no facilities are available. The Semantic Interoperability Workbench (SIW) can be used for this function.

Here are a few highlights in the meantime:

- discovery-server (eureka), config-server, edge-server (zuul) are microservice infrastructural services. At a minimum even for development environment, you would need to run config-server and discovery-server. When the microservices are fired up, they will first look for discovery-server (registry of microservice instances). After reaching the discovery-server, they will get the location of config-server (which should also have registered itself to discovery-server), register themselves and also get instance information of other microservices. Once the service information is received, the microservices can start calling each other. They will also periodically call discovery-server to see if there is any change in service registry and provide their health status. You can see the links below for more details. They are all based on Spring Cloud projects. Eureka and Zuul are originally based on Netfix OSS but we are using Spring Cloud wrappers for them.
    - https://spring.io/projects/spring-cloud-netflix [spring.io]
        - https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-eureka-server.html [cloud.spring.io]
    - https://spring.io/guides/gs/routing-and-filtering/ [spring.io]
        - https://cloud.spring.io/spring-cloud-netflix/multi/multi__router_and_filter_zuul.html [cloud.spring.io]
    - https://spring.io/projects/spring-cloud-config [spring.io]
        - https://cloud.spring.io/spring-cloud-config/multi/multi__spring_cloud_config_server.html [cloud.spring.io]
- config-server is essentially an infrastructure service that can serve microservice configuration via RESTful API in a scalable way. You can expose a folder for yml/properties file or even a git repository for configuration files. When using Spring Cloud stack, these all fluently integrate with some minimal configuration.
- Any *-ui-api project is meant as the backend for the '*' UI project. It would be basically the API meant to serve for that particular UI, so it is also needed.
- It's been a while since I worked on this application, I need to check this but you may also need to run FHIR Server even for login and may need to have certain Patient/Practitioner resources linked to the users created in UAA. I would recommend you to run all services to make sure something is not left behind. If you can create users via the UI, the FHIR resource/user links could be already established by the OCP. I suppose, if you can login with an OCP Admin user, you can start kicking things off on UI like creating organizations, organization admin users and then using those organization admins you can create patients…etc. Again, I will try to verify this information again.
- UAA is based on CloudFoundry UAA, a fork of it which we also added support for SMART on FHIR flows.
- See the application.yml files in each project, there has to be some logging.file config to write the log files for each microservice.
- When you asked about Consent2Share deployment, if you meant the SMART on FHIR app version of it that runs on OCP, then there is no other UAA is required. OCP itself is a SoF platform with UAA as its OAuth2 server. You can register and launch any SoF app on OCP. C2S SoF app believe consists of 2 projects, a UI and backend API. You just need to deploy those two and register it as a SoF app on OCP.
- As Ken mentioned, there is no reporting, data/import export feature. But as a SoF platform, its capabilities can be extended by using other SoF apps on the platform.
- We have been using docker (docker-compose) to run OCP on server environments.

Initial Areas of Discussion:

- The dozen or so different servers and their function, especially

ocp-discovery-server (Eureka), ocp-config-server,

- What parts of ocp are completely open source vs. what parts are proprietary (such as Quippe, Consent2Share)

--- only the OCP framework with care coordination and C2S are open source. Everything else that was launched is proprietary. One of the benefits of the Smart on FHIR (SoF) launch capability is that any SoF available application can be used in the OCP. Typically, it is way cheaper and better to use applications that are already built, than to try and build them from scratch.

- Patient Profile creation - confirm OCP supports Patient profile creation

- Reporting features of OCP? ---- There are no reporting features currently in OCP.

- The overall deployment process for Consent 2 Share.

Other Questions:

1) What are the minimum number of OCP servers modules to have a successful user login to the OCP UI and use most functions? Here's what we know for now -

ocp-ui, uaa-api, uaa-db. Is ocp-ui-api (and therefore ocp-config-server and ocp-discovery-server required)

2) Where can we find error logging for the OCP UI and API's?

3) What is involved with deployment of Consent 2 Share? For example we know a separate UAA is required, but what else is necessary to integrate with (OCP)?

4) What facilities for data export and import does OCP offer?

----- Currently no facilities are available.