OperatorFabric Architecture

1. Introduction

OperatorFabric is a modular, extensible, industrial-strength and field-tested platform for use in electricity, water, and other utility operations.

- System visualization and console integration
- Precise alerting
- Workflow scheduling
- Historian
- Scripting (ex: Python, JavaScript)

> Workflow Scheduling could be addressed either as an internal module or through simplified and standardized (BPMN) integration with external workflow engines, we're still weighing the pros and cons of the two options._

OperatorFabric is part of the LF Energy coalition, a project of The Linux Foundation that supports open source innovation projects within the energy and electricity sectors.

OpFab is an open source platform licensed under Mozilla Public License V2. The source code is hosted on GitHub in this repository : operatorfabric-core.

The aim of this document is to describe the architecture of the solution, first by defining the business concepts it deals with and then showing how this translates into the technical architecture.

2. Business Architecture

OperatorFabric is based on the concept of cards, which contain data regarding events that are relevant for the operator. A third party tool publishes cards and the cards are received on the screen of the operators. Depending on the type of the cards, the operator can send back information to the third party via a "response card".

2.1. Business components

To do the job, the following business components are defined :

- Card Publication : this component receives the cards from third party tools or users
- Card Consultation : this component delivers the cards to the operators and provide access to all cards exchanged (archives)
- Card rendering and process definition : this component stores the information for the card rendering (templates, internationalization, …) and a light description of the process associate (states, response card, …). This configuration data can be provided either by an administrator or by a third party tool.
- User Management : this component is used to manage users, groups and entities.

2.2. Business objects

The business objects can be represented as follows :

- Card : the core business object which contains the data to show to the user(or operator)
- Publisher : the third party which publishes or receives cards
- User : the operator receiving cards and responding via response cards
- Group : a group (containing a list of users)
- Entity : an entity (containing a list of users)
- Process : the process the card is dealing with
- State : the step in the process
- Card Rendering : data for card rendering

3. Technical Architecture

The architecture is based on independant modules. All business services are accessible via REST API.

3.1. Business components

We find here the business component seen before:

- We have a "UI" component which stores the static pages and the UI code that is downloaded by the browser. The UI is based an Angular and Handlebars for the card templating.
- The business component named "Card rendering and process definition" is at the technical level known as "Third service". This service receive card rendering and process definition as a bundle. The bundle is a tar.gz file containing
    - json process configuration file (containing states & actions)
    - templates for rendering
    - stylesheets
    - internationalization information

Except form the UI, which is based on angular, all business components are based on SpringBoot and packaged via Docker.

Spring WebFlux is used to provide the card in a fluid way.

3.2. Technical components

3.2.1. Gateway

It provides a filtered view of the APIS and static served pages for external access through browsers or other http compliant accesses. It provides the rooting for accessing the services from outside. It is a nginx server package with docker, this component contains the angular UI component.

3.2.2. Broker

The broker is used to share information asynchronously across the whole services. It is implemented via RabbitMQ

3.2.3. Authentication

The architecture provides a default authentication service via KeyCloak but it can delegate it to an external provider. Authentication is done through the use of Oauth2, three flows are supported : implicit, authorization code and password.

3.2.4. Database

The cards are stored in a MongoDb database. The bundles are stored in a file system.