

CPS-1031 - Application Properties for Configuration

✓ CPS-1031 - Review properties defined in (oom) application-helm CLOSED

- 1. Introduction
- 2. Current Implementation (Jakarta)
 - 2.1. Application Level
 - 2.2. Helm Chart Level
 - 2.3. Helm Values Level
- 3. Improvements & Cleanup
 - 3.1. Remove Unused Properties
 - 3.2. Remove Duplicated Properties
 - 3.3. Rename path for properties dynamically redefined (optional)
- 4. Example

1. Introduction

To fulfill various usage needs it is important that all CPS applications can be configured when they are deployed without requiring any change in the application artifacts (docker images).

CPS applications configuration capabilities are provided by leveraging Spring Boot application properties provided by Spring Boot:

- <https://docs.spring.io/spring-boot/docs/current/reference/html/howto.html#howto.properties-and-configuration>
- <https://www.baeldung.com/properties-with-spring#boot>

The properties that are available for configuration include both:

- Custom CPS properties specific to the application
- Spring Boot defined properties: <https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#appendix.application-properties>

2. Current Implementation (Jakarta)

CPS configuration properties are implemented at different level

2.1. Application Level

Following are the properties that should be defined at application level in the default `application.yml` file from the application repository:

- Application custom properties for any value that should not be hard-coded in the application code source. A default value is provided in the properties file. When deploying the application any user has the option to use the default value provided by the application or change the value to run with a different one.
- Spring Boot defined properties that need to be set for the application (either because Spring default value does not fit or is not set).

Detailed configuration:

- All application configuration
- Database connectivity configuration
- Kafka connectivity configuration at application level is a Plain Text basic configuration ready to be used and tested easily on a local dev environment.

For example see CPS Core:

- <https://github.com/onap/cps/blob/jakarta/cps-application/src/main/resources/application.yml>

2.2. Helm Chart Level

When CPS applications are deployed in a Kubernetes environment, they are configured to run with a specific `helm` Spring profile specified by `SPRING_PROFILES_ACTIVE` environment variable. For example, see CPS Core:

- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/templates/deployment.yaml#L92>
- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/values.yaml#L177>

Then, OOM chart `application-helm.yml` is mounted in the application container to provide `helm` profile configuration values. For example see CPS Core:

- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/templates/deployment.yaml#L102>

OOM chart `application-helm.yml` should contain following properties:

- All properties already defined in application `application.yml` file but need to be changed (overridden) to provide another default value for Kubernetes deployment and runtime.
- Any Spring Boot defined property that is not specified in application `application.yml` file but need to be specified (set) for Kubernetes deployment and runtime.

Any value property set in application `application.yml` file and not re-defined in Helm profile is still loaded and kept unchanged. There is no need to duplicate the properties in Helm profile.

Detailed configuration:

- Database connectivity configuration for with Helm defined database service values
- Kafka connectivity configuration if ONAP Strimzi Kafka is used

For example see CPS Core:

- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/resources/config/application-helm.yml>

2.3. Helm Values Level

Finally, still when deploying CPS application in a specific user environment, the user can to provide its own custom helm overrides values to customize the application in any specific target environment.

When doing this the user is still able to provide and set any additional application property he needs. Such properties are added under following properties paths:

- `config.additionalProperties.*` for any additional application or Spring property to be added
- `config.eventPublisher.*` for Kafka custom configuration properties (if ONAP Strimzi Kafka is not used)

These properties paths are automatically added to the Spring Helm profile by following Helm templating:

- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/resources/config/application-helm.yml#L66>
- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/resources/config/application-helm.yml#L62>

Detailed configuration:


- Any application configuration (custom application or default Spring Boot to be overridden or added)
- Kafka connectivity configuration if Kafka server is provisioned independently from ONAP

For example, see CPS Core:

- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/values.yml#L193>
- <https://github.com/onap/oom/blob/jakarta/kubernetes/cps/components/cps-core/values.yml#L199>

The values above are specified in the OOM CPS values file, but the user is still able to also create and provide its own overrides values file to be used when deploying.

3. Improvements & Cleanup

Jira for below work:  **CPS-1121** - Clean-up and Improve Helm Chart Properties CLOSED

3.1. Remove Unused Properties

At application level, properties that are not used can be removed:

- `spring.liquibase.labels?`
- Any other ?

3.2. Remove Duplicated Properties

At Helm chart level, considering that Helm profile is an add-on to the default profile, useless duplicates can be removed:

- `spring.liquibase.*`
- `spring.kafka.producer.client-id`
- `security.*`
- `logging.*` (and also remove `logging.*` values from values files if not used anywhere else)
- `dmi.*`

3.3. Rename path for properties dynamically redefined (optional)

Current	Proposition	Description
config. eventPublisher.*	config. kafkaConnectivity.*	for Kafka connectivity configuration
config. additional.*	config. overrides.*	Considering that all values provided at this level are overriding properties already defined or specified (either by the application itself or by Spring), it is suggested here to use only one more generic name to dynamically set these values

4. Example

Considering following configuration files for default application properties, helm profile properties and helm overrides values:

application.yaml

```
my-cps:
  property-1: my-value-1
  property-2: my-value-2
  property-3: my-value-3

spring:
  application:
    name: my-application-name
  main:
    banner-mode: console
```

application-helm.yaml

```
my-cps:
  property-2: my-other-value-2

spring:
  main:
    banner-mode: off
    log-startup-info: false
```

overrides-helm..yaml

```
config:
  overrides:
    my-cps.property-3: my-other-value-3
    spring.datasource.hikari.maximum-pool-size: 20
```

Then resulting configuration are:

- Default application running:

```
my-cps:
  property-1: my-value-1
  property-2: my-value-2
  property-3: my-value-3

spring:
  application:
    name: my-application-name
  main:
    banner-mode: console
    log-startup-info: true # Default Spring value
  datasource:
    hikari:
      maximum-pool-size: 10 # Default Spring value
```

- K8s helm deployment without overrides provided:

```
my-cps:
  property-1: my-value-1
  property-2: my-other-value-2           # From Spring Helm profile
  property-3: my-value-3

spring:
  application:
    name: my-application-name
  main:
    banner-mode: off                     # From Spring Helm profile
    log-startup-info: false              # From Spring Helm profile
  datasource:
    hikari:
      maximum-pool-size: 10              # Default Spring value
```

- K8s helm deployment with overrides provided:

```
my-cps:
  property-1: my-value-1
  property-2: my-other-value-2           # From Spring Helm profile
  property-3: my-other-value-3           # From Helm overrides

spring:
  application:
    name: my-application-name
  main:
    banner-mode: off                     # From Spring Helm profile
    log-startup-info: false              # From Spring Helm profile
  datasource:
    hikari:
      maximum-pool-size: 20              # From Helm overrides
```