Configuring of logging in Kubernetes Cluster

Information Coordinator Service

There are two ways to configure logging.

- Updating the application.yaml file, which requires a POD restart.
- By invoking a REST API. The changes takes effect immediately, but the changes will not survive a POD restart.

The contents of the log can be read by command:

```
kubectl logs informationservice-0 --namespace nonrtric
```

Updating of the configuration file application.yaml

Debug logging can be configured by a file; application.yaml. This file can in turn be updated by means of a config map named **information*service-configmap***.

The contents of the config map can be retrieved by command:

```
kubectl describe configmap informationservice-configmap -n nonrtric
```

The following lines controls the logging levels (example of the default settings):

```
logging:
  level:
    ROOT: ERROR
    org.springframework: ERROR
    org.springframework.data: ERROR
    org.springframework.web.reactive.function.client.ExchangeFunctions: ERROR
    org.oransc.ics: INFO
```

If this file is put in a directory named config; *./config/application.yaml, the config map can be updated by command:*

```
kubectl create configmap informationservice-configmap --from-file=./config --dry-run=client -n nonrtric -o yaml
| kubectl apply -f -
```

A POD restart is required for the change to take place.

Configuring logging using the REST API

Debug logging can also be configured by using REST. This does not require a POD restart. The configuration will on the other hand revert to the default after a POD restart. Here follows an example on how to enable **debug** level tracing on the whole component (all classes in the ***org.oransc.ics*** package):

*>curl http://172.17.0.7:8083/actuator/loggers/**org.oransc.ics** -i -X POST -H 'Content-Type: application/json' -d '{"configuredLevel":"**debug**"}'*

You can get the IP address and the port of the SERVICE by command:

```
kubectl get service -n nonrtric
```

Policy Management Service

There are two ways to configure logging.

- Updating the application.yaml file, which requires a POD restart.
- By invoking a REST API. The changes takes effect immediately, but the changes will not survive a POD restart.

The contents of the log can be read by command:

```
kubectl logs policymanagementservice-0 --namespace nonrtric
```

Updating of the configuration file application.yaml

Debug logging in the the PMS can be configured the same was as for the Enrichment Coordinator Service (described above). The difference is that the name of the config map is ***policymanagementservice-configmap-config*.**

Configuring logging using the REST API

In the same was for the PMS, debug logging can also be configured by using REST. This does not require a POD restart. The traces will on the other hand revert to the default after a POD restart. Here follows an example on how to enable **debug** level tracing on the whole component (all classes in the ***org.onap.ccsdk.oran.a1policymanagementservice*** package):

*>curl http://172.17.0.6:8081/actuator/loggers/**org.onap.ccsdk.oran.a1policymanagementservice** -i -X POST -H 'Content-Type: application/json' -d '{"configuredLevel":"**debug**"}'*

You can get the IP address and the port of the POD by command:

```
kubectl describe pod policymanagementservice-0 -n nonrtric
```

DMaaP Adaptor Service

In the same was as the other springboot services, there are two ways to configure logging.

- Updating the application.yaml file, which requires a POD restart.
- By invoking a REST API. The changes takes effect immediately, but the changes will not survive a POD restart.

The contents of the log can be read by command:

```
kubectl logs dmaapadapterservice-0 --namespace nonrtric
```

Updating of the configuration file application.yaml

Debug logging in the the PMS can be configured the same was as for the Enrichment Coordinator Service (described above). The difference is that the name of the config map is **dmaapadapterservice-*configmap-config*.**

Configuring logging using the REST API

In the same was for the PMS, debug logging can also be configured by using REST. This does not require a POD restart. The traces will on the other hand revert to the default after a POD restart. Here follows an example on how to enable **debug** level tracing on the whole component (all classes in the **org.oran.dmaapadapter** package):

*>curl http://172.17.0.6:8081/actuator/loggers/**org.oran.dmaapadapter** -i -X POST -H 'Content-Type: application/json' -d '{"configuredLevel":"**debug**"}'*

You can get the IP address and the port of the POD by command:

```
kubectl describe pod dmaapadapterservice-0 -n nonrtric
```