Logging with Webflux

Goal

The purpose of logging topic is about obtaining in logs all the necessary information so that the SREs can properly fulfil their duty. The target is to achieve this log information if possible: WMSV:Fulfilment Service

Challenges

Webflux

Webflux having a completely different paradigm, old models and libraries for logging with java are found to be challenging to make it work as expected. The issue lies in the fact that Webflux doesn't use the traditional thread model which all logging libraries in java were using underneath to save the logging information. So saving MDC data into the thread local memory will not be helpful as each method can be executed on a different thread with Webflux.

Webflux is the default stack chosen for commerce next platform.

However, with more work, similar experience with MDC can be achieved as well as tracing via sleuth helping libraries.

Kibana space

Kibana does not ingest log levels debug by default since it has restrictions imposed because space is expensive and while more data is ingested, the slower kibana becomes, posing a threat to all projects using it. During a flow, one request arrives, multiple requests are sent to 3rd party systems, responses arrive, and one final response is sent back to the client.

Logging the bodies of all requests/responses will prove detrimental to overall performance of the logging. Logging full requests / responses bodies should not be an option, at least for the 3rd party systems involved. Only important information needed to debug a flow is supposed to be logged.

Sanitising Requests

There are many confidential information present in headers/query parameters or request bodies. Logging has to have the possibility to sanitise all this authorisation/api-keys headers or customer emails out of the logging information.

Populate MDC context

To obtain the similar parameters, a filter within webflux, type global has to be implemented and extracted information from the request and populate all necessary parameters within MDC context will be needed. Here also the sanitisation will happen. In order to read request and perform sanitisation, a mechanism to cache the request will be needed since the default behaviour is that body can only be consumed once.