# Apache Kafka / Datix / Bettermode Integration

**1. Requirements**

**1.1. Mercedes Benz Self-Service**  
  
To use the Mercedes Benz Self-Service, the external Users (Seitwerk) needs access to the required Self-Service URLs or someone from the MB Company to manage the requirements.  
  
**1.2. Datix Requirements  
  
1.2.1. Server Access**  
To use the Mercedes Benz Services of the required Apache Kafka APIs (included / called „Datix“),  
the implementation needs access to API urls of Apache Kafka / Datix as defined in 1.2.1.1. and 1.2.1.2. as the current access is only in Mercedes Benz „INT“ environment is possible.  
  
**1.2.1.1. Bootstrap Server(s)**  
To act as producer, the implementation needs access to the Apache Kafka bootstrap URL(s), the format should be „host:port“  
  
**1.2.1.2. Schema Registry**  
To act as a consumer (event poll client), the implementation needs access to the hosted schema registry, where is interacting with the Apache Kafka environment. The Schema registry URL should be in Format „protocol://host:port“ and uses (hopefully) a “Bearer” access token with a OAUTHv2 mechanism.

**1.2.2. Access to topic, schema and group-id definition(s)**  
Apache Kafka works with different Schema definitions to reflect data structures and topics in the Schema registry. Schema definitions are registered in the Registry through the company or may be from the developer itself, who needs privilegded access to the service then.  
  
To consume objects from Apache Kafka services, the implementation as consumer (client side) requires some knowledge about the topic and the schema so the developer can deserialize objects with Apache Kafka Avro format exchange given by a GenericRecord type, a topic value and the given group-id (required to don’t get duplicated consumer events).  
  
To act as producer (send data to Apache Kafka), the implementation needs a actual schema definition (may be in Avro or JSON format) and a topic to send correct data keys and values in a object belong to a topic.  
  
  
**1.3. Implementation Requirements**  
Both the producer and consumer implementation of Apache Kafka can be very simple integrated in the existing „Community API“ spring-boot project by adding some more JAVA dependencies and  
some configuration values.

**2. Implementation**  
  
The implementaion is streight forward, there are a lot of implementation examples / tutorials  
in the internet. The range is from setup of servers to implement both producer and consumer code,  
format exchange and so on.  
  
To interact with Bettermode, there are different events (called webhooks) possible, where the   
Community API can listen to and trigger events to Apache Kafka / Datix as producer.  
  
  
**2.1. Producer implementation**  
  
To act as producer, a bean of type „KafkaProducer“ with type values of „Object“ and „GenericRecord“ can be implemented. To send a record to Kafka, the bean can be used as configured object here. The bean implementation itself is done with required configuration properties. To use the bean, the object can be integrated via CDI (constructed dependency injection) in almost every place in the code where CDI is possible.  
  
  
**2.2. Consumer Implementation**  
  
To act as consumer, there is no bean possible. Here it needs a schedule enabled spring-boot service to poll the Apache Kafka server. In my local test, it was a good idea to use (self implemented) a annotated spring-boot „Configuration“ bean with annotated methods with a GenericRecord param to have a listener for consumer records. To have different listeners, the configured „topic“ from the schema registry is used in this development pattern.  
  
  
**2.3. Local Test Environment**  
As a local test environment the developer can install a local docker image, implemented a ready to work docker composer file found here:  
  
<https://docs.confluent.io/platform/current/get-started/platform-quickstart.html>

**2.4. TEST / DEV / PROD integration**  
  
belongs to the initial requirements of server urls, schema definition and how the community-api is interacting with the „Datix“ service.  
  
  
**3. Datix platform**  
  
To use the Datix platform, may be it is needed / required to get some more documention about „how to interact with Datix from outside“, „what is the role (producer, consumer) of the Community API?“ and „what are security requirements of the Datix Platform“.