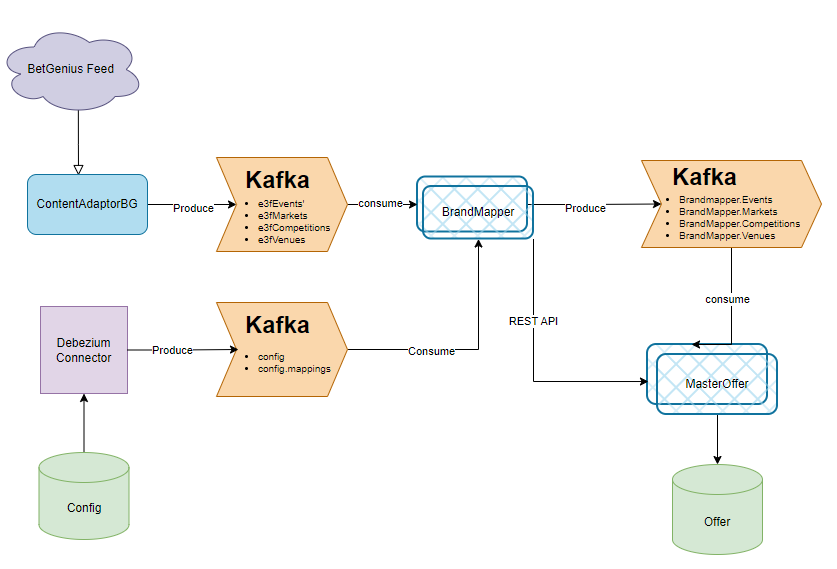
BGContentAdapter SME Documentation

# Architectural Overview

* BetGenius is the second feed introduced after BetRadar.
* BetGenius sends the feed data through REST APIs exposed by BGContentAdapter.
* The data after being transformation is added to E3F DB is put into Kafka to be consumed by BrandMapper.

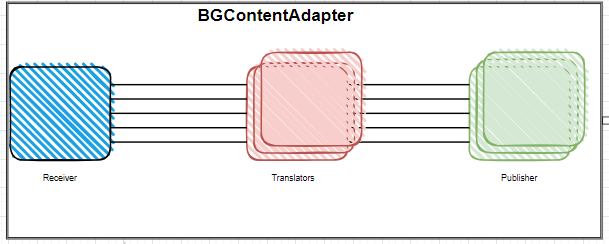


* This service isn’t horizontally scalable though it exposes REST APIs precisely because we can’t guarantee delivery of messages belonging to some event to same instance. For that we need application level load balancing currently load balancing would happen at network level.

# Internal

* Internally we are following a pipeline pattern, where a group of workers designed for specific tasks are connected through Go channels

## Receiver

* We have only one receiver which receives the BG message through REST and dispatches the message to one of the translators.
* To ensure the messages belonging to same event go to same translator(to avoid race conditions), we rely on hashing based on betgenius’ external event id.
* 

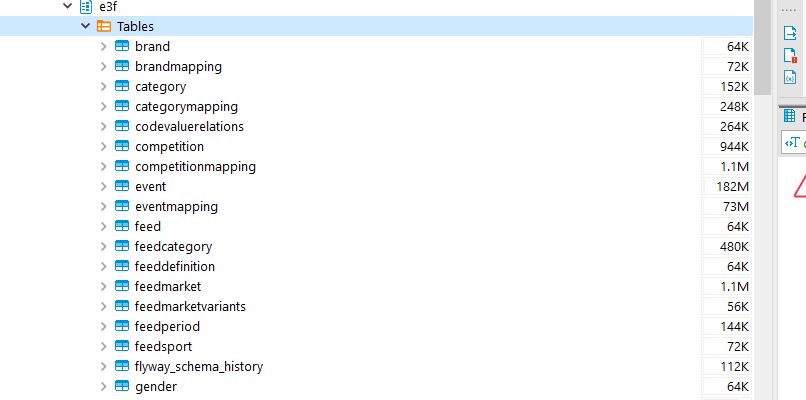
## Translator

* Number of translators and Publisher are configurable, but receiver remain’s one.
* Translator layer’s job is to convert message from BetGenius to our own internal E3F format and also store data into its E3F DB.

## Publisher

* Publisher’s job is receive message from Translator and push it to Kafka for further consumption by BrandMapper
* Number of channels remain same as number of workers as each channel connects to unique publisher.

# Tables

* 
* A screenshot of a computer

  Description automatically generated

# Kafka Topics

* env.e3fvenues
* env.e3fcompetitions
* env.e3fevents
* env.e3fmarkets

# APIs for Betgenius

/message

/heartbeat

/bbmessage

# Configs

var e3dPrefix = flag.String("e3dPrefix", "BG-", "Prefix for e3d Id")  
  
var dataReceivers = flag.String("receivers", "betgenius", "list of receivers")  
  
var betgeniusFeedDefinitionID = flag.String("receiver.betgenius.feedDefinitionId", "BETGENIUS", "feed definition id of betgenius provider")  
var betgeniusReceivingPort = flag.String("receiver.betgenius.receiver.port", "", "port of betgenius http server")  
var betgeniusDelayMs = flag.Int64("receiver.betgenius.delayms", 25000, "betgenius delay in millis for suspending offer")  
  
var e3fReceiverKafkaBrokers = flag.String("receiver.e3f.kafka.brokers", "", "CSV list of kafka broker IP:PORT")  
var e3fReceiverKafkaGroupName = flag.String("receiver.e3f.kafka.groupName", "", "group name of kafka receiver")  
var e3fPublisherKafkaBrokers = flag.String("publisher.e3f.kafka.brokers", "", "CSV list of kafka broker IP:PORT")  
var kafkaVersion = flag.String("kafka.version", "", "kafka API version")  
  
var dbName = flag.String("db.name", "", "name of database")  
var dbHost = flag.String("db.host", "", "host where db is located")  
var dbPort = flag.String("db.port", "", "port on whitch database is listening")  
var dbUser = flag.String("db.user", "", "db user")  
var dbPass = flag.String("db.pass", "", "password for database")  
  
var cacheSize = flag.Int("cache.size", 10000, "size of the cache")  
  
var dataPublishers = flag.String("publishers", "", "list of publishers")  
  
var e3WsURL = flag.String("publisher.e3ws.url", "", "location of e3d service")  
  
var controllerTranslatorThreads = flag.Int("controller.translatorthreads", 10, "number of goroutines servicing translationg")  
var controllerMaxReceivedNotTranslated = flag.Int("controller.maxreceivednottranslated", 100, "number of max received not translating messages")  
var controllerMaxReceivedNotTranslatedIdleMs = flag.Int("controller.maxreceivednottranslatedidlems", 100, "idle time after max received not translating messages reached")  
var controllerE3WsPublisherThreads = flag.Int("controller.e3ws.publisherthreads", 10, "number of go routines servicing e3ws publishing")  
var controllerE3WsPublisherBufferSize = flag.Int("controller.e3ws.publisherbuffersize", 100, "buffer size of controller publisher queues")  
  
var port = flag.String("health.port", ":8080", "Listening port")  
var internalAPIsPort = flag.String("internalapis.port", ":8880", "Internal APIs listening port")  
  
var loggingTime = flag.Bool("logging.logtime", false, "log time with every message")  
  
var dummyRun = flag.Bool("dummy", false, "Dummy service run: no publishing, no database write")  
  
var monitoringNotificationsEnabled = flag.Bool("monitoring.notificationsenabled", false, "monitoring notifications enabled")  
  
var e3fRecoveryPeriodLimit = flag.Int("e3f.recovery.period", 60, "period between two e3f recovery requests for same id")  
  
var coverageflag = flag.Bool("coverage.flag", true, "coverage flag for enabling coverage implementation")  
  
var restrictMarketWithDuplicateName = flag.Bool("duplicateMarketName.flag", false, "duplicateMarketName flag enable the markets with the duplicate name for an event")