**Data Adapter**

**Automated Testing Strategy**

# 

# **Introduction**

Automated testing must be defined for Data Adapter. Validations are required for the following items:

* Output messages
* Integration with Connect
* Performance

# **Output messages validation**

1. Validations on every message:
   1. Existence of mandatory fields
   2. Expected format for every field
2. Validations on configuration messages:
   1. Validate configuration entities comparing with TE
   2. For each entity validate:
      1. Existence of mandatory fields
      2. Expected format for every field
3. Validations on events messages:
   1. Validate AssetId and EventCaptureDate fields
   2. Validate events comparing with TE
   3. For each event validate:
      1. Existence of mandatory fields
      2. Expected format for every field

# **Integration with Connect**

Validate the output of Data Adapter is not enough, also the interaction with Connect must be validated. Since Data Adapter implementation tends to be more dependent on Connect's implementation as the time passes, the necessity of integration tests is imminent.

The following validations are required:

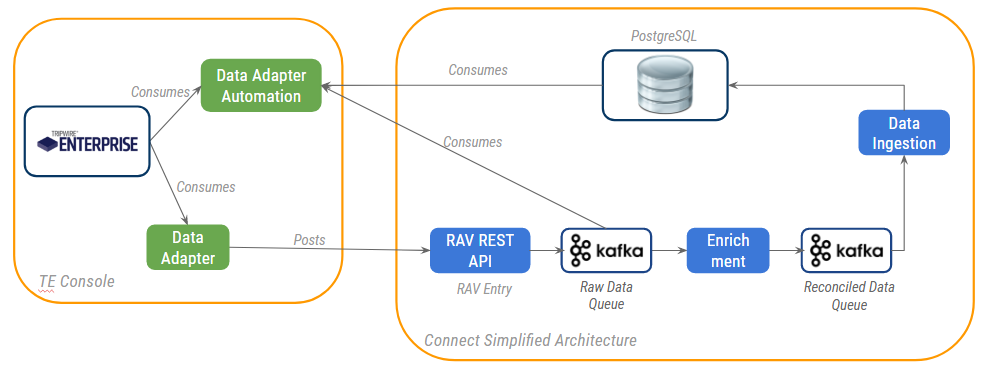
1. Datasource installation
2. Watermark posting
3. Status posting

# **Performance**

Performance can be easily tested from Data Adapter’s side by using the measures log. But, it could be also useful to measure performance taking into account the arriving time for the messages to Connect’s landing zones.

# **Proposed automated solution**

A good ‘Test Tool’ (name must be changed), should be able to extract data from TE to compare with the produced messages. Those messages can be easily extracted from PostgreSQL landing table. If desired, the messages can be extracted from the Kafka queues, but it will require a more sophisticated implementation. Also, the configuration for the data source, the watermarks and the last status can be easily extracted from PostgreSQL tables.



# **Current Implementation**

At the moment, the project is a simple Groovy application that must be executed after DA. When executed, the application runs all the available tests, returning the results for each one on the standard output. Any framework, library or gradle task will be added as needed.

## Libraries

* Groovy
* Google Reflections library - used to collect test annotations
* PostgreSQL Connector

## Implemented Tests

* Verify correct installation, validating the content of data\_source table in PostgreSQL with the info obtained from the /status endpoint of TE REST API.
* Verify last configuration message headers, validating the content or format of the attributes. The messages are extracted from scm.message\_landing table in PostgreSQL.
* Verify latest events messages headers, validating the content or format of the attributes. The messages are extracted from scm.message\_landing table in PostgreSQL.
* Verify latest events messages AssetID and EventCaptureDate headers, validating that every event on the a message has those attributes.

# **Suggested Improvements**

* Add run reports, counting scanned events and configuration messages
* Externalize configuration
* Run DA from gradle task
* Reset Connect data from gradle task
* Add missing tests
* Extract information about DA execution from measures.csv file
  + Use to create complex tests that validate watermarks
  + Use to create complex tests that validate events content
* Check Kafka queues