Catalog Data API Design

1. Background / Introduction

*This design is intended to assist in developing APIs for Catalog (Product, SaleItem/SKU, Image, Inventory and Price) Feeds. Depending on Standards or Training, the definition of a Product differs. The detail design on "how to expose APIs" is available at "**Backend Integration Architecture and Design**". This design, however, tries to dig into the sources where Products and their attributes including prices, currently available. The details about endpoints & requests/responses are available in Swagger documentation at "API Documents".*

2. Use Cases Affected

ToDo: Should be mapped back with the Use Case for a functional requirement.

*"*INT-19, 20, 22*" (Based on the frequency and size of the data gets updated, catalog feed is thoughtfully split into four feeds as Product, Price, Image and Inventory) -*

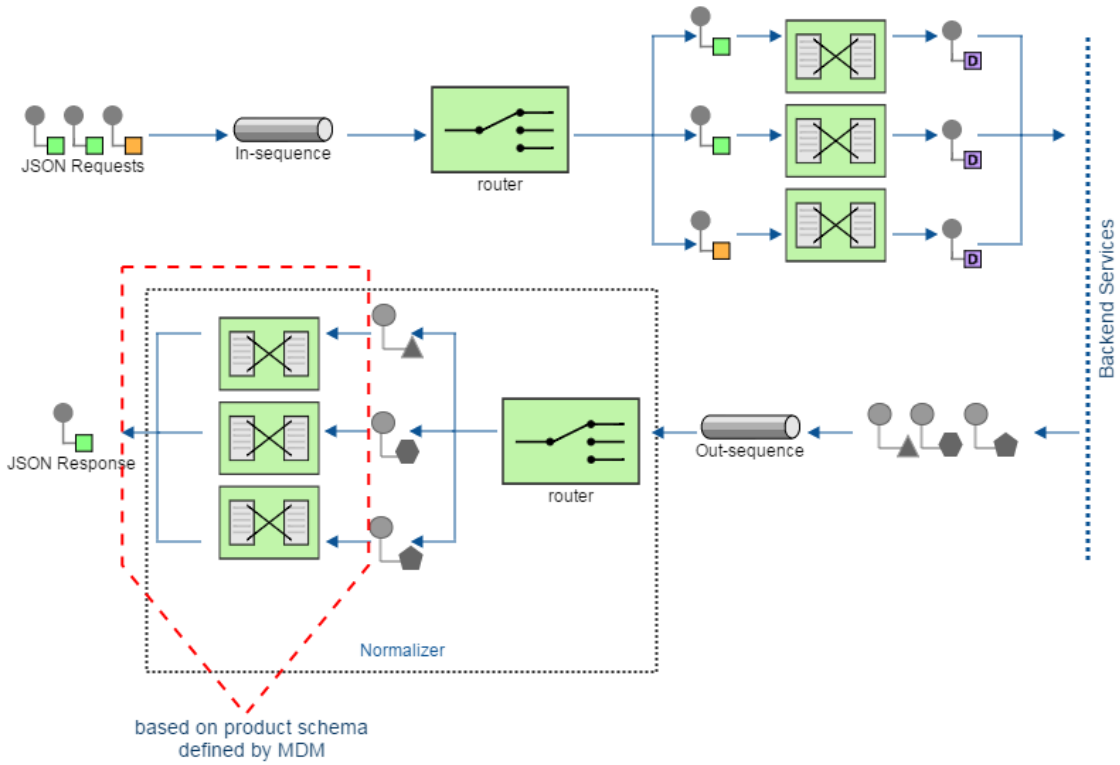
- *Product Feed ("INT-19 -* TIS Course Catalog Feed*", "INT-20 - Product Feed")*
- *Price Feed (based on store code e.g. TIS, Infostore etc)*
- *Image Feed*
- *Inventory Feed ("INT-22 - TIS Class Inventory - training", and standards inventory mainly for HC,SB..)*

3. Design Details

Catalog data feed is a data integration which addresses the requirement to update data in Intershop with the data in EPCatalogue. The data flow is initiated by Intershop by invoking Catalog Feeds endpoint. Based on store (Infostore, TIS, NSAI or BRC XI), the process flow might be different depending on backend services. For example, for TIS store, it can be SOQL based backend while for Infosrore, it's JDBC based. Having said that, backend implementation should not have any impact on consumer. All the underlying backend implications should be nicely interfaced by ESB.

For example, REST API for product feeds - for TIS, it gets "Training" products from Salesforce (backend service) using SOQL while for Infostore, it gets "Standards" products from EPCatalogue (DB) using JDBC connectivity. But from API consumer perspective - consumers require responses in JSON using the same protocol and same API. So technically speaking, this scenario would contain-

- Message - REST Message
- Message Channel - Data type channel
- Message Routing - checking store TIS vs Infostore
- Message Translator - Protocol switching and normalizer - SOAP/JDBC to REST
- Message Transformation - Data conversion - JSON



4. Endpoints

See Swagger documentation for more details about Endpoints, Request and Response formats. All the endpoints should allow partial, delta and chunks. This design covers the following endpoints -

- */products*
- */products/{productId}*

5. Request Message Format

All the above endpoints are GET requests with STORE_CODE and AUTH_CODE in header. The request would look like as below -

- *"HTTP Method: GET /products"*
- *"HTTP Method: GET /products/delta?from=<DATE>&to=<DATE>"*
- *"HTTP Method: GET /products?fields=ProductID,Publisher,Designation"*
- *"HTTP Method: GET /products?range=0-399"*

- *"HTTP Method: GET /products/{id}"*
- *"HTTP Method: GET /products/{id}/delta?from=<DATE>&to=<DATE>"*
- *"HTTP Method: GET /products/{id}?fields=ProductID,Publisher,Designation"*
- *"HTTP Method: GET /products/{id}?range=0-399"*

6. Response Message Format

The response would be JSON data. As of now, the response would be based on the EPCatalogue database schema and Object definition in Salesforce.

The response format should be decided by Information Management Techniques like Master Data Management (MDM). Once MDM is in place and provides a better product definition, then response format would be changed and should look similar to one below in the block -

```
[
   {
      "id":1,
      "name":"",
      "description":"",
      .
      .
   .
      "skus":[
         saleItem1,
         saleItem2
      ],
      "availability":"available"
   },
   {
      "id":2,
      "name":"",
      "description":"",
      .
      .
   .
      "skus":[
         saleItem3,
         saleItem4
      ],
      "availability":"available"
   }
]
```

7. Configuration

config files -

8. Security

For all REST API requests, authentication is done by verifying token over HTTPs in Authorization header. The token can be created (or use an existing one) by accessing WSO2 IS. After token is acquired, it would be passed in request header for WSO2 IS to validate/verify.

Balance between Security & Performance - As catalog REST APIs are backend APIs and would be invoked by another trusted system (Intershop) hence API can relatively be less restricted/secured. Security Architect can think of some other options like 1.) using just TLS or 2.) No-expiry token over HTTP or 3.) Just HTTP with no token and TLS.

9. Process Flow

This section will define end-to-end flow of the integration and also about if any validation, enrichment, transformation, routing. This section should be reviewed by the Tech Lead or Architect and Business Analyst.

10. Backend Services

- Infostore Standards(Products) API Design
- TIS Training (Product) API Design

11. Reference Documents