2. API Design & Response Standard

2.1. Mục Tiêu Chương

Chương này quy định cấu trúc chuẩn của mỗi integration flow và trách nhiệm của từng artifact layer nhằm:

- Tách biệt rõ ràng các tầng xử lý trong luồng tích hợp
- Giảm phụ thuộc giữa các thành phần artifact
- Dễ bảo trì, mở rộng và kiểm thử từng layer độc lập
- Đảm bảo tính nhất quán giữa các flow trong toàn hệ thống

2.2. Khái Niệm / Phạm Vi Áp Dụng

Quy tắc này áp dụng cho:

- Tất cả các integration flow trong thư mục `artifacts/**`
- Cả **Core Team** và **Partner Team**
- Tất cả các luồng tích hợp mới được phát triển

Cấu Trúc Artifact Chuẩn

Mỗi integration flow gồm các thành phần sau:

```
artifacts/
  apis/                          ← REST API layer (điểm vào từ client)
  sequences/
    <flow>-inboundSequence.xml     ← Mediation layer (xử lý logic)
    <flow>-inboundErrorSequence.xml ← Error handling layer
  inbound-endpoints/             ← Consumer layer (nhận message từ Kafka)
  local-entries/                 ← Connection & config layer (dùng chung)
  endpoints/                     ← Backend endpoint layer
```

Ý Nghĩa Từng Artifact Layer

Layer	Artifact	Vai trò	Ví dụ
API Layer	`apis/*.xml`	Nhận request từ client và trả response	KafkaProducerApi.xml, PlanningDirectApi.xml
Mediation Layer	`sequences/*-inboundSequence.xml`	Xử lý logic, điều phối luồng	Load_balance_example-inboundSequence.xml
Error Layer	`sequences/*-inboundErrorSequence.xml`	Xử lý và cô lập lỗi	Load_balance_example-inboundErrorSequence.xml
Consumer Layer	`inbound-endpoints/*.xml`	Nhận message từ Kafka topic	Load_balance_example.xml
Connection Layer	`local-entries/*.xml`	Cấu hình kết nối dùng chung	KafkaConnection.xml
Endpoint Layer	`endpoints/*.xml`	Định nghĩa backend endpoint	Backend API address

2.3. Quy Định Chính

Trách Nhiệm Từng Layer

API Layer (`apis/`)

- Nhận request HTTP từ client (hoặc từ WSO2 APIM Gateway)
- Gọi backend trực tiếp hoặc publish message lên Kafka
- Trả response chuẩn về client
- **Không** chứa logic nghiệp vụ phức tạp

Mediation Layer (`sequences/*-inboundSequence.xml`)

- Xử lý message nhận từ Kafka consumer
- Điều phối lời gọi đến backend API
- Phân loại kết quả theo HTTP status code (2xx / 4xx / 5xx)
- Publish kết quả vào `processed_topic` hoặc `error_topic`
- **Không** xử lý lỗi kỹ thuật (dành cho Error Layer)

Error Layer (`sequences/*-inboundErrorSequence.xml`)

- Bắt lỗi kỹ thuật khi consume message từ Kafka
- Log đầy đủ thông tin: `ERROR_CODE`, `ERROR_MESSAGE`, `ORIGINAL_PAYLOAD`
- Đóng gói và publish message lỗi vào `error_topic`
- **Không** retry — chỉ ghi nhận và chuyển sang error topic

Consumer Layer (`inbound-endpoints/`)

- Kết nối đến Kafka broker và subscribe topic
- Khai báo `sequence` (luồng thành công) và `onError` (luồng lỗi)
- **Không** chứa logic xử lý

Connection Layer (`local-entries/`)

- Lưu thông tin kết nối Kafka dùng chung toàn hệ thống
- Chỉ Core Team được phép chỉnh sửa

Quy Tắc Bắt Buộc

> **API Layer không được chứa logic nghiệp vụ phức tạp**

> **Mọi response trả về client phải theo cấu trúc JSON chuẩn**

> **Error sequence phải luôn publish message vào `error_topic`**

> **Mediation sequence phải phân loại HTTP status (2xx / 4xx / 5xx)**

Cấu Trúc Response Chuẩn

Thành công:

```
{
  "success": true,
  "message": "Hồ sơ đã được gửi thành công"
}
```

Thành công kèm data:

```
{
  "eventType": "transaction",
  "source": "<sequence-name>",
  "data": { ... }
}
```

Lỗi từ API Layer:

```
{
  "success": false,
  "httpStatus": "400",
  "error": { ... }
}
```

Lỗi kỹ thuật (Error Sequence ghi vào `error_topic`):

```
{
  "errorSource": "Load_balance_example-inboundEndpoint",
  "sourceTopic": "test_topic_01",
  "errorType": "KAFKA_CONSUME_ERROR",
  "errorCode": "...",
  "errorMessage": "...",
  "originalPayload": { ... },
  "timestamp": "..."
}
```

2.4. Cách Thực Hiện / Quy Trình

Quy Trình Phát Triển Một Integration Flow Mới

Bước 1: Khai báo Connection (nếu chưa có)

Tạo hoặc tái sử dụng `local-entries/KafkaConnection.xml`.

> Chỉ Core Team thực hiện bước này.

Bước 2: Tạo Inbound Endpoint

Tạo file `inbound-endpoints/<flow-name>.xml`:

- Khai báo Kafka broker, topic, group ID
- Trỏ `sequence` đến inbound sequence
- Trỏ `onError` đến error sequence

Bước 3: Tạo Error Sequence

Tạo file `sequences/<flow-name>-inboundErrorSequence.xml`:

- Log đầy đủ `ERROR_CODE`, `ERROR_MESSAGE`, `ORIGINAL_PAYLOAD`
- Build payload lỗi chuẩn
- Publish vào `error_topic`

Bước 4: Tạo Inbound Sequence

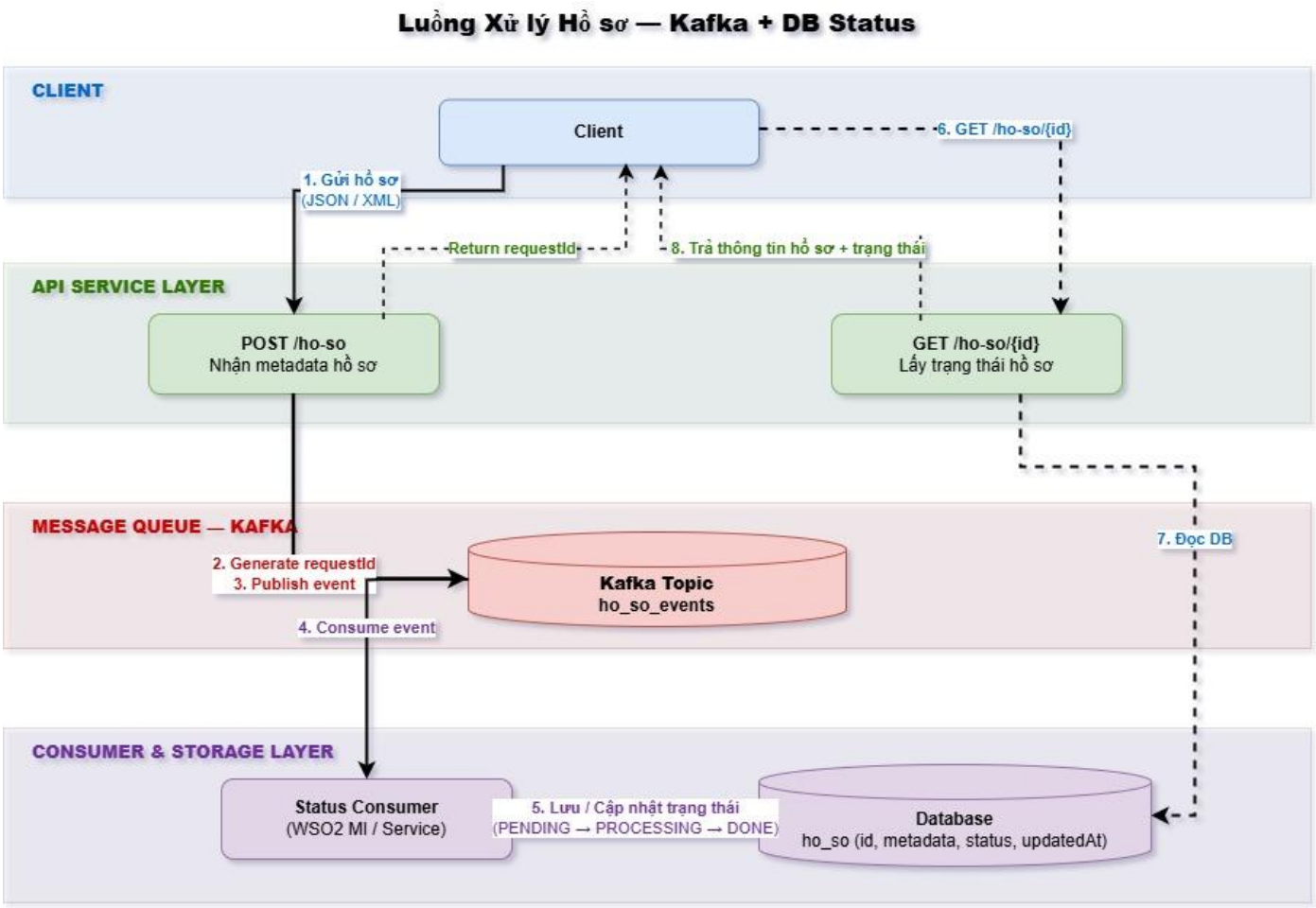
Tạo file `sequences/<flow-name>-inboundSequence.xml`:

- Log message nhận được
- Set header xác thực
- Gọi backend API (`<call>`)
- Phân loại theo HTTP status:
  - `200 / 201` → publish vào `processed_topic`
  - `4xx` → đóng gói lỗi dữ liệu → publish vào `error_topic`
  - `5xx` → đóng gói lỗi backend → publish vào `error_topic`

Bước 5: Tạo API (nếu cần điểm vào HTTP)

Tạo file `apis/<flow-name>Api.xml`:

- Khai báo context path và method
- Set header, gọi backend hoặc publish Kafka
- Trả response chuẩn theo HTTP status



2.5. Ví Dụ Minh Họa

Ví Dụ Đúng — Inbound Sequence phân loại HTTP Status

```
<!-- Thành công: publish vào processed_topic -->
<filter xpath="get-property('HTTP_STATUS') = '200' or get-property('HTTP_STATUS') = '201'">
   <then>
      <log level="custom">
         <property name="STATUS" value="[KafkaConsumer][SUCCESS] Backend xử lý thành công"/>
      </log>
      <payloadFactory media-type="json" template-type="default">
         <format>{
            "eventType": "transaction",
```

```
            "source": "Load_balance_example-inboundSequence",
            "data": ${payload}
          }</format>
      </payloadFactory>
      <kafkaTransport.publishMessages configKey="KafkaConnection">
         <topic>processed_topic</topic>
      </kafkaTransport.publishMessages>
   </then>
   <else>
      <!-- Lỗi: publish vào error_topic -->
      <kafkaTransport.publishMessages configKey="KafkaConnection">
         <topic>error_topic</topic>
      </kafkaTransport.publishMessages>
   </else>
</filter>
```

Đúng vì:

- Phân loại rõ 2xx (thành công) và lỗi
- Kết quả thành công đi vào `processed_topic`
- Lỗi được tách riêng vào `error_topic`
- Sequence không xử lý lỗi kỹ thuật (để cho Error Sequence)

Ví Dụ Đúng — Error Sequence đóng gói lỗi chuẩn

```
<sequence name="Load_balance_example-inboundErrorSequence">
   <!-- Bước 1: Lưu thông tin lỗi -->
   <property name="ERROR_CODE"      expression="get-property('ERROR_CODE')"    scope="default"/>
   <property name="ERROR_MESSAGE"   expression="get-property('ERROR_MESSAGE')" scope="default"/>
   <property name="ORIGINAL_PAYLOAD" expression="json-eval($)"                 scope="default"/>

   <!-- Bước 2: Log chi tiết -->
   <log level="custom">
      <property name="STATUS"         value="[KafkaConsumer][ERROR] Lỗi khi xử lý message"/>
      <property name="ERROR_CODE"     expression="get-property('ERROR_CODE')"/>
      <property name="ERROR_MESSAGE"  expression="get-property('ERROR_MESSAGE')"/>
   </log>
```

```
    <!-- Bước 3: Đóng gói payload lỗi -->
    <payloadFactory media-type="json">
        <format>{
            "errorSource": "Load_balance_example-inboundEndpoint",
            "sourceTopic": "test_topic_01",
            "errorType":   "KAFKA_CONSUME_ERROR",
            "errorCode":   "$1",
            "errorMessage":"$2",
            "originalPayload": $3
        }</format>
    </payloadFactory>

    <!-- Bước 4: Publish vào error_topic -->
    <kafkaTransport.publishMessages configKey="KafkaConnection">
        <topic>error_topic</topic>
    </kafkaTransport.publishMessages>
</sequence>
```

Ví Dụ Sai — Không phân loại HTTP Status

```
<!-- SAI: Không kiểm tra status, trả về luôn -->
<call>
    <endpoint>
        <address uri="http://192.168.0.133:8080/api/v1/plannings"/>
    </endpoint>
</call>
<respond/>
```

Sai vì:

- Không kiểm tra HTTP status code từ backend
- Lỗi 4xx / 5xx không được phân loại, không vào `error_topic`
- Không log trạng thái xử lý

Ví Dụ Sai — API trả về raw data không theo chuẩn

```
<!-- SAI: Không wrap response theo chuẩn -->
<payloadFactory media-type="json">
   <format>{"data": "ok"}</format>
</payloadFactory>
<respond/>
```

Sai vì:

- Response không nhất quán với chuẩn `success / error` toàn hệ thống

2.6. Checklist Áp Dụng

Trước khi commit integration flow mới:

- ☐ Có đủ: `inbound-endpoint`, `inboundSequence`, `inboundErrorSequence`
- ☐ Inbound Sequence có phân loại HTTP status (2xx / 4xx / 5xx)
- ☐ Thành công publish vào `processed_topic`
- ☐ Lỗi publish vào `error_topic`
- ☐ Error Sequence log đầy đủ: `ERROR_CODE`, `ERROR_MESSAGE`, `ORIGINAL_PAYLOAD`
- ☐ Error Sequence đóng gói payload lỗi theo cấu trúc chuẩn
- ☐ API Layer trả response theo cấu trúc JSON chuẩn (`success`, `error`)
- ☐ Không hardcode thông tin kết nối trong Sequence (dùng `local-entries`)
- ☐ PR đã được Core Team review

*Tài liệu này thuộc phạm vi quản lý của **Core Team** — mọi thay đổi phải được Core Team phê duyệt.*

Phiên bản #5
Được tạo 2026-02-23 07:50:10 UTC bởi Admin
Được cập nhật 2026-02-25 04:01:38 UTC bởi Nam Đặng