3.Cơ chế xác thực (JWT/Bearer)

3.1. Mục Tiêu Chương

Chương này quy định chuẩn xác thực và nguyên tắc thiết kế bảo mật nhằm:

- Đảm bảo cơ chế xác thực thống nhất giữa các integration flow và đơn vị phát triển
- Giúp artifact dễ đọc, dễ hiểu và dễ bảo trì
- Giảm lỗi phát sinh do xử lý token không đúng chuẩn
- Tăng hiệu quả review và kiểm soát chất lượng artifact
- Đảm bảo toàn bộ luồng từ Client → APIM → MI → Backend đều có xác thực nhất quán
- Tích hợp đúng với WSO2 Identity Server (IS) làm trung tâm cấp và xác thực token

3.2. Khái Niệm / Phạm Vi Áp Dụng

Quy định này áp dụng cho:

- Toàn bộ artifact xử lý hoặc chuyển tiếp request qua xác thực
- **Core Team** và **Partner Team**
- Tất cả các API, Sequence, Inbound Endpoint trong `artifacts/**`
- Cấu hình APIM và IS do Core Team quản lý

Kiến Trúc Xác Thực Tổng Thể

Ba Tầng Xác Thực

Tầng	Thành phần	Vai trò xác thực
Cấp token	WSO2 Identity Server (IS)	OAuth2 Authorization Server, cấp JWT/Access Token
Validate token	WSO2 API Manager (APIM)	Xác thực token tại Gateway trước khi route vào MI
Forward token	WSO2 Micro Integrator (MI)	Set `Authorization: Bearer` header khi gọi backend

3.3. Quy Định Chính

3.3.1. Quy Tắc Đặt Tên Artifact

Các artifact liên quan xác thực phải đặt tên theo quy ước thống nhất:

Thành phần	Quy tắc đặt tên	Ví dụ
REST API	`<Name>Api.xml`	KafkaProducerApi.xml, PlanningDirectApi.xml
Inbound Sequence	`<flow>-inboundSequence.xml`	Load_balance_example-inboundSequence.xml

Error Sequence	`<flow>-inboundErrorSequence.xml`	Load_balance_example-inboundErrorSequence.xml
Inbound Endpoint	`<flow>.xml`	Load_balance_example.xml
Connection Entry	`<Name>Connection.xml`	KafkaConnection.xml
Backend Endpoint	`<Name>Endpoint.xml`	`PlanningBackendEndpoint.xml`

3.3.2. Quy Tắc Đặt Property Trong Sequence

Dùng tên property nhất quán khi xử lý token/header:

```
<!-- Đúng: Tên property rõ ràng, đúng chuẩn -->
<property name="PAYLOAD"          expression="json-eval($)"    scope="default"/>
<property name="HTTP_STATUS"      expression="$axis2:HTTP_SC"  scope="default"/>
<property name="ERROR_CODE"       expression="get-property('ERROR_CODE')"    scope="default"/>
<property name="ERROR_MESSAGE"    expression="get-property('ERROR_MESSAGE')" scope="default"/>

<!-- Sai: Tên không mô tả ý nghĩa -->
<property name="x"    expression="json-eval($)"   scope="default"/>
<property name="code" expression="$axis2:HTTP_SC" scope="default"/>
```

3.3.3. Quy Định Về Logging Xác Thực

Không được bỏ qua bước log khi xử lý request:

```
<!-- Sai: Không log trạng thái -->
<call>
   <endpoint>
      <address uri="http://192.168.0.133:8080/api/v1/plannings"/>
   </endpoint>
</call>
```

Phải log đầy đủ các bước:

```
<!-- Đúng: Log đầy đủ theo chuẩn dự án -->
<log level="custom">
   <property name="API_NAME" value="PlanningDirectApi"/>
   <property name="STATUS"   value="Nhận request, gọi thẳng Backend..."/>
   <property name="PAYLOAD"  expression="get-property('PAYLOAD')"/>
</log>
```

3.3.4. Quy Định Về Token / Header Xác Thực

Token dùng để gọi backend phải được set từ `local-entries` hoặc config.properties — **không hardcode trong Sequence**:

```
<!-- SAI: Hardcode token trong Sequence -->
<header name="Authorization" scope="transport"
   value="Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiJ9..."/>
```

```
<!-- ĐÚNG: Lấy token từ property/config -->
<property name="BACKEND_TOKEN"
   expression="get-property('conf:backend.auth.token')"
   scope="default"/>
<header name="Authorization" scope="transport"
   expression="fn:concat('Bearer ', get-property('BACKEND_TOKEN'))"/>
```

3.4. Nguyên Lý Tách Biệt Trách Nhiệm (Separation of Concerns)

Tương đương nguyên lý **SOLID** áp dụng trong hệ thống WSO2:

3.4.1. S — Mỗi Artifact Một Trách Nhiệm

Mỗi artifact chỉ được thực hiện một nhiệm vụ:

Artifact	Chỉ được làm
`apis/`	Nhận HTTP request, set header auth, gọi backend/Kafka, trả response
`sequences/*-inboundSequence.xml`	Điều phối luồng message từ Kafka đến backend
`sequences/*-inboundErrorSequence.xml`	Ghi nhận lỗi và publish vào `error_topic`
`inbound-endpoints/`	Kết nối Kafka, subscribe topic
`local-entries/`	Lưu thông tin kết nối và cấu hình dùng chung

Sai — Error logic nằm trong inbound sequence:

```
<!-- SAI: Inbound sequence tự xử lý lỗi kỹ thuật -->
<sequence name="myFlow-inboundSequence">
    <call>...</call>
    <property name="ERROR_CODE" expression="get-property('ERROR_CODE')"/>
    <kafkaTransport.publishMessages configKey="KafkaConnection">
        <topic>error_topic</topic>
    </kafkaTransport.publishMessages>
    <!-- Đây là nhiệm vụ của inboundErrorSequence -->
</sequence>
```

Đúng — Error sequence riêng biệt:

```
<inboundEndpoint
    sequence="myFlow-inboundSequence"
    onError="myFlow-inboundErrorSequence">
```

3.4.2. O — Mở Rộng Không Sửa Artifact Cũ

Khi thêm topic/luồng mới, tạo artifact mới — không sửa sequence của luồng khác:

```
<!-- Thêm luồng mới: tạo file riêng -->
<!-- planning_topic-inboundSequence.xml -->
<!-- planning_topic-inboundErrorSequence.xml -->
<!-- planning_topic.xml (inbound endpoint) -->
```

3.4.3. L — Consumer Tuân Thủ Contract Sequence

Inbound Endpoint phải trỏ đúng sequence và error sequence đã khai báo:

```
<!-- Sai: onError trỏ sai sequence -->
<inboundEndpoint name="Load_balance_example"
   sequence="Load_balance_example-inboundSequence"
   onError="someOtherSequence"/>  <!-- không tồn tại -->
```

```
<!-- Đúng -->
<inboundEndpoint name="Load_balance_example"
   sequence="Load_balance_example-inboundSequence"
   onError="Load_balance_example-inboundErrorSequence"/>
```

3.4.4. I — Tách Riêng Local Entry Theo Chức Năng

Không nhét tất cả cấu hình vào một local entry:

```
<!-- Sai: Một local entry chứa tất cả -->
<localEntry key="AllConfig">
   <!-- Kafka config, backend URL, token, ... -->
</localEntry>

<!-- Đúng: Tách riêng -->
<localEntry key="KafkaConnection">...</localEntry>
<localEntry key="BackendConfig">...</localEntry>
```

3.4.5. D — Phụ Thuộc Config, Không Hardcode

Sequence phụ thuộc vào `local-entries` và config.properties — không hardcode URL hay token:

```
<!-- Sai -->
<address uri="http://192.168.0.133:8080/api/v1/plannings"/>
```

```
<!-- Đúng -->
<address uri="{$ctx:backend.planning.url}"/>
```

3.5. Luồng Xác Thực Chuẩn Trong Hệ Thống

3.5.1. Luồng Chuẩn Từ Client Đến Backend

```
Client
  → [1] Lấy token từ WSO2 IS (/oauth2/token)
  → [2] Gọi API qua WSO2 APIM với Bearer token
  → [3] APIM validate token với WSO2 IS (introspect / JWT verify)
  → [4] APIM route request vào WSO2 MI
  → [5] MI set header Authorization khi gọi backend
  → [6] Backend (Spring Boot) validate JWT
```

3.5.2. Luồng Kafka Consumer Không Đi Qua APIM

```
Kafka Topic (test_topic_01)
  → [1] MI Inbound Endpoint consume message
  → [2] MI Inbound Sequence xử lý
  → [3] MI set header Authorization: Bearer <token>
  → [4] MI gọi Backend API
  → [5] Backend validate JWT
  → [6] Kết quả: published_topic hoặc error_topic
```

> **Luồng Kafka consumer bypass APIM Gateway — token phải được quản lý thủ công trong MI (lấy từ IS hoặc dùng service account token).**

3.5.3. Không Được Phép

```
<!-- Sai: Gọi backend không có Authorization header -->
<call>
   <endpoint>
      <address uri="http://192.168.0.133:8080/api/v1/plannings"/>
   </endpoint>
</call>

<!-- Sai: API tại APIM không bật Security Scheme -->
<!-- Mọi API publish lên APIM đều phải bật OAuth2 hoặc API Key -->
```

3.5.4. Luồng Token Trong Kafka Consumer (Chuẩn)

```
<!-- Bước 1: Lấy token từ config -->
<property name="BACKEND_TOKEN"
   expression="get-property('conf:backend.auth.token')"
   scope="default"/>

<!-- Bước 2: Set Authorization header -->
<header name="Authorization" scope="transport"
   expression="fn:concat('Bearer ', get-property('BACKEND_TOKEN'))"/>
<property name="Content-Type" value="application/json" scope="transport"/>

<!-- Bước 3: Gọi backend -->
<call>
   <endpoint>
      <address uri="{$ctx:backend.planning.url}">
         <suspendOnFailure>
            <initialDuration>1000</initialDuration>
            <progressionFactor>2.0</progressionFactor>
            <maximumDuration>60000</maximumDuration>
         </suspendOnFailure>
      </address>
   </endpoint>
```

```
</call>
```

3.6. Cấu Hình Xác Thực WSO2 APIM & IS

3.6.1. WSO2 IS — Key Manager

WSO2 IS đóng vai trò **Key Manager** cho APIM:

Chức năng	Endpoint WSO2 IS
Lấy Access Token	`POST /oauth2/token`
Introspect Token	`POST /oauth2/introspect`
Lấy Public Key (JWKS)	`GET /oauth2/jwks`
Revoke Token	`POST /oauth2/revoke`

Lấy token từ IS (client_credentials):

```
POST https://<IS_HOST>:9443/oauth2/token
Content-Type: application/x-www-form-urlencoded
Authorization: Basic <base64(clientId:clientSecret)>

grant_type=client_credentials&scope=<scope>
```

3.6.2. WSO2 APIM — API Gateway Security

Mọi API publish lên APIM phải cấu hình:

Thuộc tính	Giá trị chuẩn
Security Type	OAuth2 hoặc API Key
Token Validation	IS Introspection / JWT verify
Throttling	Theo chính sách dự án
CORS	Cấu hình theo môi trường

3.6.3. JWT Claims Chuẩn Của Hệ Thống

Token JWT cấp bởi WSO2 IS phải chứa:

```
{
  "sub": "admin",
  "role": "ADMIN",
  "iat": 1771917707,
  "exp": 1772004107,
  "iss": "https://<IS_HOST>:9443/oauth2/token",
  "aud": "...client_id...",
  "scope": "..."
}
```

3.7. Cách Thực Hiện / Quy Trình

Quy Trình Thêm Xác Thực Cho Luồng Mới

Bước 1: Đăng ký OAuth2 Application trong WSO2 IS

- Tạo Service Provider
- Lấy `clientId` và `clientSecret`

Bước 2: Cấu hình token trong

```
backend.auth.token=<access_token_từ_IS>
backend.planning.url=http://192.168.0.133:8080/api/v1/plannings
```

Bước 3: Trong Sequence, đọc token từ config

```
<property name="BACKEND_TOKEN"
    expression="get-property('conf:backend.auth.token')"/>
<header name="Authorization" scope="transport"
    expression="fn:concat('Bearer ', get-property('BACKEND_TOKEN'))"/>
```

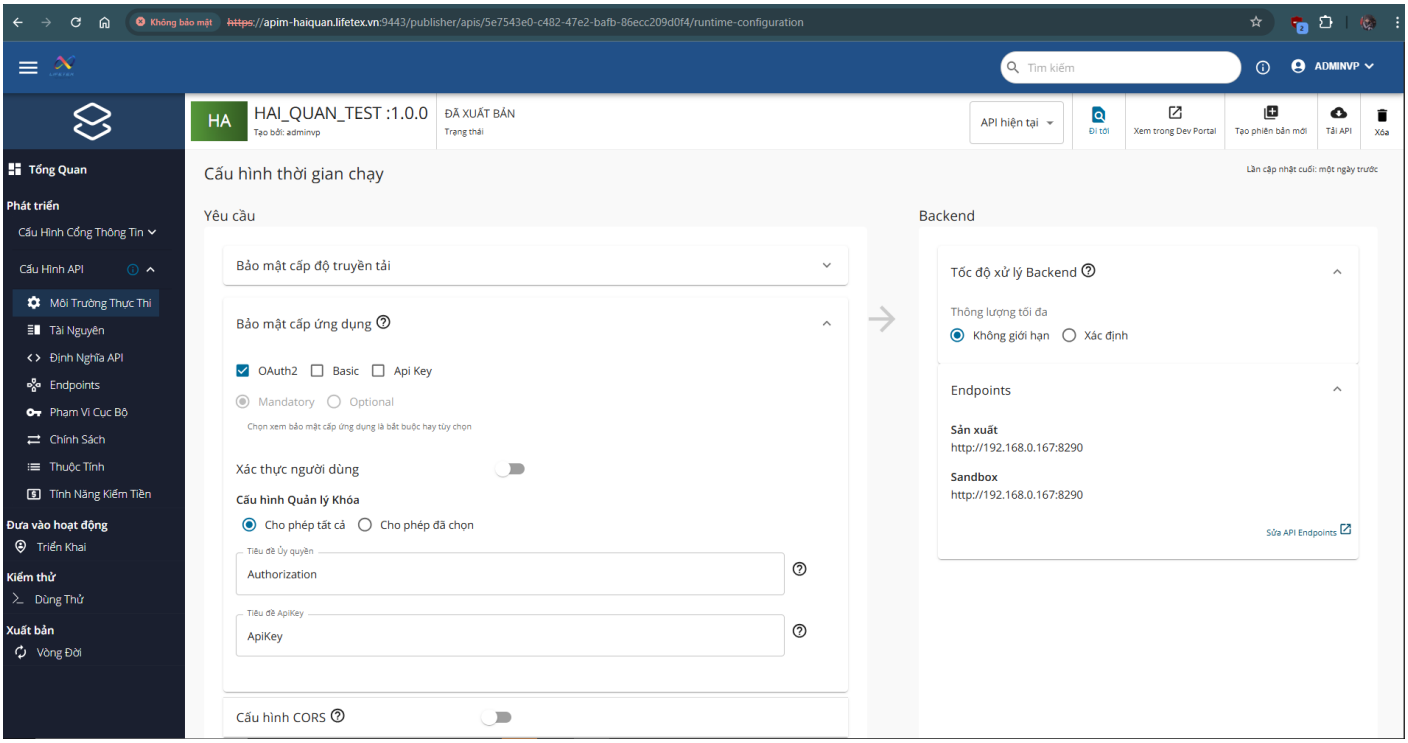
Bước 4: Publish API lên APIM với OAuth2 Security

- Import OpenAPI spec từ `resources/api-definitions/`
- Enable Security: **OAuth2**

- Set Key Manager: **WSO2 IS**

Bước 5: Test luồng xác thực end-to-end

- Lấy token từ IS
- Gọi API qua APIM với Bearer token
- Verify log trong MI: `STATUS`, `HTTP_STATUS`, `PAYLOAD`



3.8. Ví Dụ Minh Họa

Ví Dụ Sai — Sequence Không Có Authorization Header

```
<sequence name="myFlow-inboundSequence">
   <call>
      <endpoint>
         <address uri="http://192.168.0.133:8080/api/v1/plannings"/>
      </endpoint>
   </call>
   <respond/>
</sequence>
```

Sai vì:

- Không set `Authorization: Bearer` header
- Không log request/response
- Backend sẽ reject với 401 Unauthorized
- Không xử lý HTTP status response

Ví Dụ Sai — Hardcode Token Trong Sequence

```
<header name="Authorization" scope="transport"
   value="Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiJ9..."/>
```

Sai vì:

- Token hardcode sẽ hết hạn, phải sửa code mỗi lần
- Lộ thông tin xác thực trong source code
- Không thể thay đổi theo môi trường (dev / staging / prod)

Ví Dụ Đúng — Sequence Xác Thực Và Gọi Backend Chuẩn

```
<sequence name="Load_balance_example-inboundSequence">

   <!-- 1. Lưu payload gốc -->
   <property name="PAYLOAD" expression="json-eval($)" scope="default"/>
   <log level="custom">
      <property name="STATUS"  value="[KafkaConsumer] Message nhận được từ Kafka"/>
      <property name="TOPIC"   value="test_topic_01"/>
      <property name="PAYLOAD" expression="get-property('PAYLOAD')"/>
   </log>

   <!-- 2. Set Authorization header từ config -->
   <property name="BACKEND_TOKEN"
      expression="get-property('conf:backend.auth.token')" scope="default"/>
   <header name="Authorization" scope="transport"
      expression="fn:concat('Bearer ', get-property('BACKEND_TOKEN'))"/>
   <property name="Content-Type" value="application/json" scope="transport"/>
```

```
    <!-- 3. Gọi backend -->
    <call>
        <endpoint>
            <address uri="{$ctx:backend.planning.url}">
                <suspendOnFailure>
                    <initialDuration>1000</initialDuration>
                    <progressionFactor>2.0</progressionFactor>
                    <maximumDuration>60000</maximumDuration>
                </suspendOnFailure>
            </address>
        </endpoint>
    </call>

    <!-- 4. Kiểm tra phản hồi -->
    <property name="HTTP_STATUS"   expression="$axis2:HTTP_SC"  scope="default"/>
    <property name="API_RESPONSE"  expression="json-eval($)"    scope="default"/>
    <log level="custom">
        <property name="STATUS"      value="[KafkaConsumer] Phản hồi Backend"/>
        <property name="HTTP_STATUS" expression="get-property('HTTP_STATUS')"/>
    </log>

    <!-- 5. Phân loại kết quả -->
    <filter xpath="get-property('HTTP_STATUS') = '200' or get-property('HTTP_STATUS') = '201'">
        <then>
            <kafkaTransport.publishMessages configKey="KafkaConnection">
                <topic>processed_topic</topic>
            </kafkaTransport.publishMessages>
        </then>
        <else>
            <kafkaTransport.publishMessages configKey="KafkaConnection">
                <topic>error_topic</topic>
            </kafkaTransport.publishMessages>
        </else>
    </filter>

</sequence>
```

> **[Chỗ này thêm ảnh: WSO2 MI log viewer — hiển thị STATUS, HTTP_STATUS sau khi gọi backend thành công]**

Ví Dụ Đúng — API APIM Với OAuth2 Security

```
# resources/api-definitions/KafkaProducerApi.yaml
openapi: 3.0.0
info:
  title: Kafka Producer API
  version: 1.0.0
paths:
  /kafka-producer:
    post:
      summary: Publish message lên Kafka
      security:
        - OAuth2: []          # Bắt buộc xác thực OAuth2 tại APIM
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        '200':
          description: Thành công
components:
  securitySchemes:
    OAuth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://<IS_HOST>:9443/oauth2/token
          scopes: {}
```

3.9. Checklist Áp Dụng

Trước khi commit hoặc tạo PR:

Naming

- ☐ API artifact: `<Name>Api.xml`
- ☐ Inbound Sequence: `<flow>-inboundSequence.xml`
- ☐ Error Sequence: `<flow>-inboundErrorSequence.xml`
- ☐ Connection Entry: `<Name>Connection.xml`

Token & Xác Thực

- ☐ Không hardcode Bearer token trong Sequence
- ☐ Token lấy từ

config.properties hoặc `local-entries`

- ☐ Mọi lời gọi backend đều có `Authorization: Bearer` header
- ☐ API publish lên APIM đã bật OAuth2 Security
- ☐ Key Manager trỏ đúng về WSO2 IS

Logging

- ☐ Sequence log `STATUS`, `PAYLOAD` khi nhận request
- ☐ Sequence log `HTTP_STATUS`, `API_RESPONSE` sau khi gọi backend
- ☐ Error Sequence log `ERROR_CODE`, `ERROR_MESSAGE`, `ORIGINAL_PAYLOAD`

Separation of Concerns

- ☐ API Layer không chứa logic điều phối phức tạp
- ☐ Error handling nằm trong Error Sequence riêng
- ☐ Kết nối Kafka trong `local-entries`, không khai báo lại trong Sequence
- ☐ URL backend lấy từ config, không hardcode

Architecture

- ☐ Inbound Endpoint trỏ đúng `sequence` và `onError`
- ☐ Luồng thành công publish vào `processed_topic`

- [ ] Luồng lỗi publish vào `error_topic`
- [ ] Không có luồng nào bỏ qua xác thực với backend

3.10. Cấu Trúc Chuẩn Của Một Sequence Xử Lý Nghiệp Vụ

3.10.1. Mục Tiêu

Quy định cấu trúc thống nhất của một `inboundSequence` nhằm:

- Dễ đọc và dễ review
- Tách rõ: **nhận dữ liệu → xác thực → gọi backend → phân loại kết quả**
- Tránh thiếu log hoặc thiếu xử lý HTTP status
- Chuẩn hóa cách publish kết quả vào Kafka topic

3.10.2. Cấu Trúc Bắt Buộc

Một `inboundSequence` phải theo thứ tự:

```
1. Lưu payload gốc vào property
2. Log thông tin message nhận được
3. Set Authorization header (Bearer token)
4. Gọi backend API (<call>)
5. Lưu HTTP status và response
6. Log phản hồi backend
7. Phân loại theo HTTP status (filter)
   - 2xx → publish vào processed_topic
   - 4xx → đóng gói lỗi dữ liệu → publish vào error_topic
   - 5xx → đóng gói lỗi backend → publish vào error_topic
```

3.10.3. Quy Định Chi Tiết Từng Bước

1. Lưu payload:

```
<property name="PAYLOAD" expression="json-eval($)" scope="default"/>
```

2. Log input:

```
<log level="custom">
   <property name="STATUS"  value="[KafkaConsumer] Message nhận được từ Kafka"/>
   <property name="PAYLOAD" expression="get-property('PAYLOAD')"/>
</log>
```

3. Set header xác thực (từ config, không hardcode):

```
<property name="BACKEND_TOKEN"
   expression="get-property('conf:backend.auth.token')"/>
<header name="Authorization" scope="transport"
   expression="fn:concat('Bearer ', get-property('BACKEND_TOKEN'))"/>
```

4. Gọi backend:

```
<call>
   <endpoint>
      <address uri="{$ctx:backend.planning.url}">
         <suspendOnFailure>
            <initialDuration>1000</initialDuration>
            <progressionFactor>2.0</progressionFactor>
            <maximumDuration>60000</maximumDuration>
         </suspendOnFailure>
      </address>
   </endpoint>
</call>
```

5–6. Lưu và log phản hồi:

```
<property name="HTTP_STATUS"  expression="$axis2:HTTP_SC"  scope="default"/>
<property name="API_RESPONSE" expression="json-eval($)"    scope="default"/>
```

```
<log level="custom">
   <property name="HTTP_STATUS"    expression="get-property('HTTP_STATUS')"/>
   <property name="RESPONSE_BODY"  expression="get-property('API_RESPONSE')"/>
</log>
```

7. Phân loại kết quả:

```
<filter xpath="get-property('HTTP_STATUS') = '200' or get-property('HTTP_STATUS') = '201'">
   <then>
      <!-- publish processed_topic -->
   </then>
   <else>
      <filter xpath="get-property('HTTP_STATUS') >= '400' and get-property('HTTP_STATUS') &lt; '500'">
         <then>
            <!-- 4xx: lỗi dữ liệu → error_topic -->
         </then>
         <else>
            <!-- 5xx: lỗi backend → error_topic -->
         </else>
      </filter>
   </else>
</filter>
```

3.10.4. Ví Dụ Sequence Sai

Sai vì: Không có Bearer token, không log, không phân loại HTTP status, không publish kết quả.

3.10.5. Checklist Review Sequence

Khi review một `inboundSequence`:

- ☐ Có lưu `PAYLOAD` vào property trước khi xử lý
- ☐ Có log thông tin nhận message
- ☐ Có set `Authorization: Bearer` header (từ config, không hardcode)
- ☐ Có log HTTP status sau khi gọi backend

- [ ] Có phân loại HTTP status: 2xx / 4xx / 5xx
- [ ] 2xx → `processed_topic`
- [ ] 4xx và 5xx → `error_topic`
- [ ] Không xử lý lỗi kỹ thuật trong sequence này (để cho Error Sequence)

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

Phiên bản #6
Được tạo 2026-02-23 07:50:11 UTC bởi Admin
Được cập nhật 2026-02-26 02:48:24 UTC bởi Nam Đặng