**PAYMENT SERVICE**

**Game Integration Documents**

**Version: 1.8**

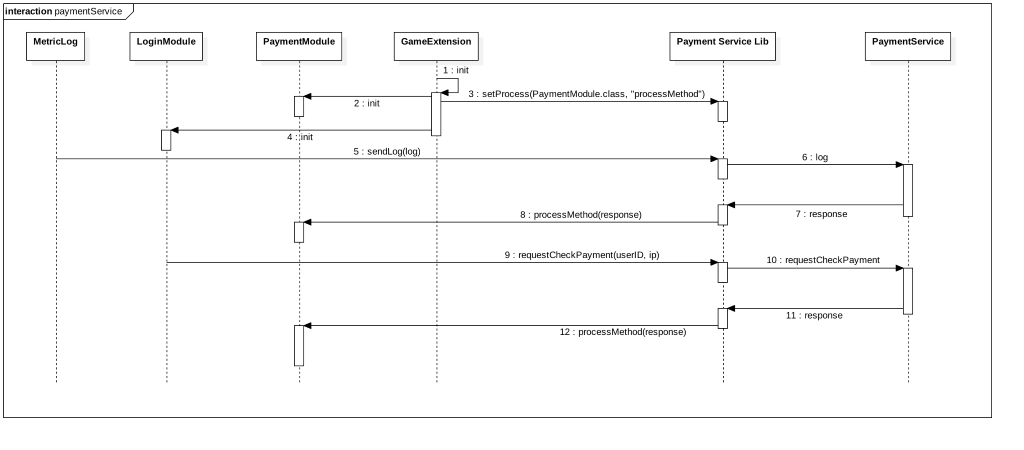
Revision History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0 | 19/12/2018 | - Draft documents |
| 1.1 | 22/02/2019 | * Update flow and demo |
| 1.2 | 3/4/2019 | Update list connection |
| 1.3 | 19/4/2019 | Update some note |
| 1.4 | 2/5/2019 | Update VN connection |
| 1.5 | 8/5/2019 | Update config multiple server |
| 1.6 | 21/5/2019 | Thêm quy trình tích hợp |
| 1.7 | 6/6/2019 | Thêm note java version, thêm thông tin countryID |
| 1.8 | 17/9/2019 | Thêm hướng dẫn thêm keystore |

1. **Mô tả**

Payment Service là service quản lý việc hiển thị trạng thái, các kênh nạp tiền trong game. Game sẽ cung cấp một số thông tin cần thiết cho Service thông qua Kafka Streaming, phía Service sẽ xử lý và trả về kết quả realtime ngay khi có đủ thông tin cung cấp từ phía game

1. **Flow hoạt động**



Khi user đăng nhập hoặc có sự thay đổi trong data, game sẽ gửi thông tin lên Service. Service sẽ trả về kết quả trong 2 trường hợp

* Tự động khi có đủ thông tin và trạng thái payment của user thay đổi
* Khi có request yêu cầu cung cấp trạng thái từ phía game

1. **Tích hợp trong game**
2. **Demo**

Xem thư mục [integrate](https://vngms-my.sharepoint.com/:f:/r/personal/hieupt_vng_com_vn/Documents/GSN/PaymentService/integrate?csf=1&e=llEkKk)

Các lib cần thiết để tích hợp Broker:

* Kafka clients 2.0 trở lên
* Gson
* Json
* log4j-1.2.15
* Slf4j-api-1.6.1
* slf4j-log4j12-1.6.1
* Spymemcached-2.8.9
* common lang 3

***NOTE 1:*** Ngoài ra, lib và config trên private được share trên super-dev folder #congnt2/paymentService

***NOTE 2***: Lưu ý, nên sử dụng các lib có version >= các lib trong demo để đảm bảo tính ổn định cho lib service, tránh crash game

***NOTE 3*:** lib chỉ hỗ trợ ***java8*** trở lên, các game cần chú ý update version java trước khi tích hợp.

1. **Step by step**
2. Khởi tạo

Trong DemoExtention.init(), gọi hàm sau

|  |  |
| --- | --- |
| DemoExtension.init() | |
| Broker.getInstance().setProcess(PaymentModule.class, “processMethod”); |  |

Trong đó “processMethod” là hàm xử lý Response, hàm này có thể được các game đặt tên khác nhau, chỉ cần config đúng tên để payment lib có thể tìm được và gửi response về socket.

**Lưu ý:** Class xử lý response phải được thiết kế theo pattern singleton và có hàm static getInstance().

1. Send log

Các thông tin cần thiết sẽ được gửi dưới dạng log như ghi log metric, nhưng thay vì dùng Scribe sẽ dùng Kafka. Một số thông tin cần thiết:

|  |  |
| --- | --- |
| **Logname** | **Description** |
| UserId | id user trong game |
| Username | username |
| MCC | 5 ký tự, bao gồm MCC và MNC code, tham khảo https://goo.gl/KjBpCW |
| appName | Distribution của phiên bản (Google, Marketing...) |
| appVersion | appVersion của phiên bản (dạng INT) |
| Level | Level (nếu có) của user |
| Win | Số trận thắng (nếu có) |
| Lost | Số trận thua (nếu có) |
| Install date | Ngày bắt đầu cài đặt (format: “dd-mm-yyyy") |
| IP | IP user (ipv4) |
| DeviceID | DeviceID của user |
| Platform | Android, iOS,… |
| CountryID | Country ID của user |

Nếu trong LogMetric có sẵn các thông tin này thì chỉ cần cung cấp lại format log cho phía Service và ghi thêm vào Broker như ví dụ trong MetricLog.writeActionLogNew(). Còn nếu không sẽ cùng định nghĩa lại MetricLog cho một số trường còn thiếu.

1. Request

|  |  |
| --- | --- |
| Request | |
| Function  Broker.getInstance().sendCheckPayment(userID, userIP);  Params:  {  userID : individual id for game  userIP : IP of user  } |  |

1. Response

|  |  |
| --- | --- |
| Response | |
| Function  PaymentModule.processMethod(message)  Params:  {  message: response trả về, format: uId|isEnable   * uId: id của user * isEnable: boolean, trạng thái đóng / mở payment   } |  |

1. LogTest

|  |
| --- |
| logTest |
| Function  Broker.getInstance().logTest(userId, gameLogicResult, serviceResult)  Params:  {  userId: id của user  gameLogicResult: Kết quả enable payment phía gameLogic  userId: Kết quả enable payment phía service  } |

Để logTest, cần config category TestEnablePayment trong log4j

1. Config

Xem trong thư mục “conf”

Một số config bắt buộc:

* gameId: String. Đây ID của game, config này sẽ được thống nhất giữa service và game.
* useCache: Boolean. Config này quyết định có cache lại thông tin payment của user hay không. Nếu useCache = true, cần config thêm thông tin về database, config này sẽ được đề cập sau. Ngoài ra, còn có thể sử dụng hàm getEnablePayment(String uId) để lấy ra thông tin payment của user từ cache. Nếu useCache = false, hàm trên sẽ trả về null. ***Lưu ý:*** *Để sử dụng tính năng cache này, lib memcached phải có version >= lib đi kèm với payment service (hiện tại đang là 2.8.9). Hiện tại, một số product đang sử dụng memcached-2.5 nếu bật tính năng này sẽ không nhận được response từ phía service.*
* serverIndex*:* int. Chỉ số của game server trong trường hợp game có nhiều server. Nếu game chỉ có 1 server có thể bỏ qua config này. (Nên đánh số liên tiếp bắt đầu từ 0).
* Database: JSONObject. Config này chứa thông tin về database dùng để cache thông tin payment của user. Nếu useCache = false, config này coi như không có giá trị. Config có dạng:

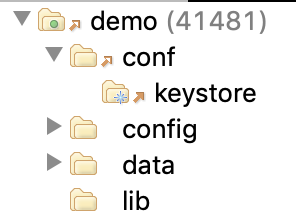
“database”: { “host”: “hostIP”, “port”: “port”}

Các config khác sẽ được thống nhất trong quá trình tích hợp.

1. Lưu ý

* Broker được thiết kế theo Singleton pattern nên trong quá trình tích hợp và sử dụng cần chú ý.
* Để xem log, thêm module Broker vào log4j.
* Các lib cần thiết nằm trong phần demo. Nếu lib phía game đang dùng phiên bản cũ hơn mà không update theo lib đi kèm, service có thể không hoạt động đúng theo mong muốn.
* Nên config useCache = true để hạn chế việc gọi lại nhiều lần, đồng thời giảm rủi ro khi có sự cố mạng hay phía service.

1. Danh sách connection
   * Private:
     + 10.30.44.108:9092,9093
     + 10.30.44.109:9092,9093
     + 10.30.44.107:9092,9093
   * Live:
     + SEA:
       - Kafka1.zingplay.com:9092, 9093
       - Kafka2.zingplay.com:9092, 9093
       - Kafka3.zingplay.com:9092, 9093
     + VN:
       - kafka1-new.zingplay.com:9092, 9093
       - kafka2-new.zingplay.com:9092, 9093
       - kafka3-new.zingplay.com:9092, 9093
2. **Quy trình tích hợp**
   * Game gửi rule ẩn hiện payment hiện tại (nếu có)
   * Thống nhất format log mới hoặc dùng lại log đã có (game gửi kèm format log hiện tại)
   * Tích hợp – check connection
     + Game check connect đến cụm 3 server private như mục III.2.h, nếu không được thì request acl đến cụm trên
     + Nếu server game live nằm ở VN thì check và request acl đến cụm live VN. Hiện tại cụm service đang nằm ở newFarm, nếu game đang nằm ở newFarm thì request qua đường private, còn nếu đang ở oldFarm thì request qua đường public
     + Nếu service live nằm ở SEA thì request acl đên cụm live SEA
     + Nên config useCache = true để hạn chế rủi ro
     + Database có thể dùng chung với database của game
   * Sau khi đã tích hợp và thông kết nối đến private sẽ tiến hành test trên dev
   * Sau khi kết quả test đã tốt sẽ tiến hành up live sau khi thông kết nối live
3. **Hướng dẫn thêm keystore**

****Payment service sử dụng 2 key gồm 1 truststore và 1 keystore. Giả sử folder deploy có cấu trúc như sau:

Thư mục con/keystore chứa các key. Vậy config các location như sau:

* “truststoreLocation”: “conf/keystore/truststore.jks”
* “keystoreLocation”: “conf/keystore/keystore.jks”

Hoặc nếu các key không đặt trong folder deploy thì cần config đường dẫn tuyệt đối từ thư mục gốc “/”.