PATIGAMES Server API 0.4

시스템 사용 전 협의 사항

파티게임즈에서 제공하는 관리페이지에 게임이 등록되어 있는지, SDK/API 메뉴에서 gameid / ClientData(manifest에 추가하는 encryptedClientData) / serverKey등이 발급되어 있는지 확인한다.

관리 페이지 : https://admin.patigames.com(라이브) / https://admintest.patigames.com (스테이지)

API 호출 공통 사항

이 API는 게임 서버에서 호출하도록 설계되었다.

API 호출 방식은 모두 POST이며, POST DATA, RESPONSE DATA는 모두 JSON 형식이다.
RESPONSE DATA는 모두 object 형식이며, object 속 code가 0 이상이면 성공이고, 그렇지 않으면 에러 코드이다.

> 성공 예: {"code":0,"result": ... }
> 실패 예: {"code":-900, "msg":"malformed parameter"}

유효한 호출인지 검증하기 위해 HTTP헤더에 HMAC-MD5를 추가한다. HMAC key는 serverKey이고, 해시 알고리즘은 MD5를 사용한다. serverKey가 0123456789abcdef이고, {"test":"message"}인 경우의 코드와 헤더 예제는 아래와 같다.

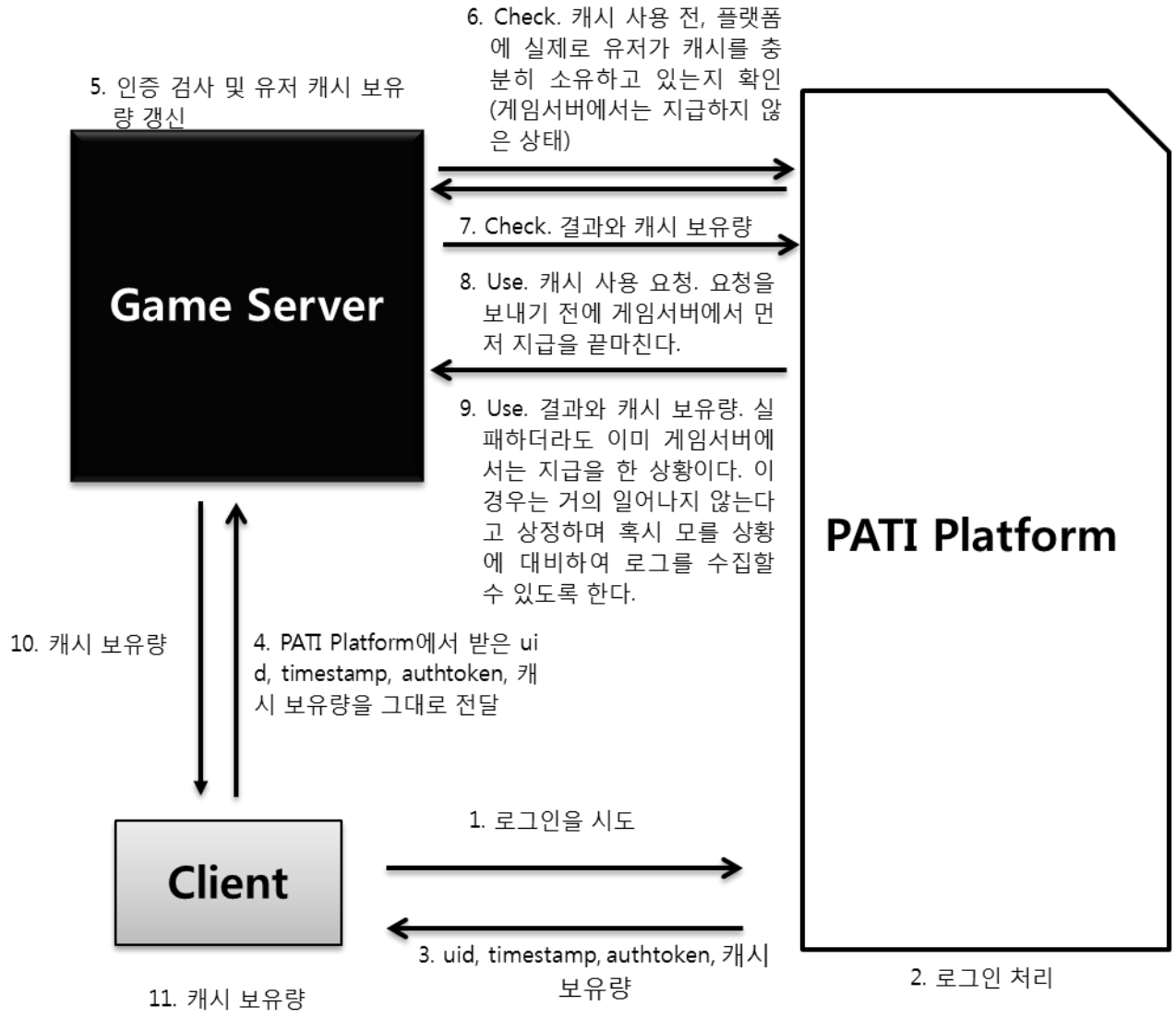
```
POST /api/call HTTP/1.1
Host: sapi.patigames.com
Connection: Keep-Alive
Cache-Control: no-cache
HMAC-MD5: a13213e3b3f9df838773e5720bbdc62a

{"test":"message"}
```

API 서버 주소

- Test : http://sapitest.patigames.com/ or https://sapitest.patigames.com/
- Live : http://sapi.patigames.com/

로그인 검증 / 캐시 동기화 프로세스



로그인 검증

클라이언트가 보낸 uid, timestamp와 미리 발급된 serverKey의 조합으로 계산한 값이 클라이언트에서 보낸 authToken과 같아야 하고(다르면 해킹), timestamp가 서버시간과 차이가 크다면 에러로 간주해야 한다.

(*) serverkey는 client에 절대 노출하면 안되며, 게임 서버만 알고 있어야 한다.

이런 프로세스는 악의적인 유저가 Client를 해킹하거나, 가짜 클라이언트를 만들어 다른 사람의 계정을 해킹하는 문제를 방지하기 위해 꼭 필요하다. (이런 절차를 거치지 않으면 게임 서버는 해당 uid가 PATI Platform에서 정상적인 프로세스로 온 것이 맞는지 확인할 수가 없다.)

[인증 검사 방법]

1. 클라이언트가 보낸 uid, timestamp와 미리 발급된 serverkey의 조합으로 계산한 값이 클라이언트가 보낸 authtoken과 일치하는지 확인한다. 채널을 사용하는 경우 channelnum을 포함해야 한다.

```
EX) if (authTokenFromClient != sha256( str(uid) + str(timestamp) + str(serverKey))

EX2) if (authTokenFromClient != sha256( str(uid) + '%30d' % channelnum + str(timestamp) + str(serverKey))

        throw Exception("잘못된 로그인: 해킹시도 가능성 있음");
```

2. 클라이언트가 보낸 timestamp를 현재 서버시간과 비교해 차이가 크다면 조작되었을 가능성이 있으므로 에러로 간주해야 한다. (timestamp는 GMT 기준 unix_timestamp이다.)

[* 언어별 sha256 사용법]

(sha256('hello world!') = 7509e5bda0c762d2bac7f90d758b5b2263fa01ccbc542ab5e3df163be08e6ca9)

1. PHP : `echo hash('sha256', 'hello world!');`
2. Python :
```
import hashlib
print hashlib.sha256("hello world!").hexdigest()
```

캐시 동기화

플랫폼에서는 네트워크를 통한 트랜잭션 등 사용한 캐시를 다시 롤백하는 수단을 제공하고 있지 않다. 따라서 다음과 같이 게임서버에서 캐시를 동기화 하는 것을 권장한다.

1. 기본적으로 서버는 캐시를 사용하기 전 항상 플랫폼에 해당 유저가 캐시를 충분히 보유하고 있는지 확인한다.
2. 캐시 보유량이 충분함을 확인 후 실제로 캐시 사용요청을 보내기 전에 게임서버에서 먼저 캐시 사용에 따른 보상을 지급한다.
3. 캐시 사용에 따른 보상 지급이 완전히 마무리(변경사항을 DB에까지 완전히 저장) 된 후에 플랫폼으로 캐시 사용 요청을 보낸다.
4. 이렇게 하면 플랫폼에서 캐시가 차감되었는데 게임에서 지급이 되지 않는 문제는 발생하지 않게 된다. (유저가 손해 볼 수 있는 케이스는 제거) 대신 게임서버에서 지급되었는데 플랫폼에서 캐시가 차감이 되지 않는 경우는 발생할 수 있다. 따라서 로그 등을 통해 후속 조치할 여지를 남겨 둘 필요는 있다.
5. 1의 이유로 서버는 실질적으로 유저의 캐시 보유량을 알고 있을 필요는 없다. 그러나 캐시 보유량을 서버 메모리 상에 올려놓고 검사하고 싶다면 최초의 값 설정은 클라이언트에서 플랫폼 로그인 후 받아온 캐시 보유량을 사용한다. (로그인 검증과 같은 단계이므로 신뢰할 수 있음. 또한 유저 로그인마다 플랫폼으로 요청하여 생기는 네트워크 부하를 줄일 수 있음)

/s/<gameid>/cash/check , /s/<gameid>/<channelnum>/cash/check

유저의 보유 캐시량이 사용하려는 캐시량 이상인지 확인한다. 잔액이 충분하면 성공, 모자라면 실패 처리된다. 어느 케이스에든 현재 유저의 캐시 잔액이 함께 내려온다. (아예 잔액을 조회할 수 없는 에러가 존재하기 때문에 실패의 케이스에는 paid, bonus, free가 없는 경우 현재의 값을 유지할 수 있도록 하는 조치가 필요하다)

이 API는 /cash/use와 쌍을 이루며 캐시를 사용하려는 시작시점에 항상 호출되어야 한다. 서버가 구현해야 할 캐시 사용 흐름은 다음과 같다.

잔액 갱신 및 사용 가능 확인 -> 캐시 사용 대가 지급 -> 캐시 사용

API 호출의 결과를 확인하고 다음 과정을 진행해야 하므로 위 과정은 동기화되어야 한다. 또한 캐시를 사용하기 이전에 항상 캐시 잔액을 갱신하게끔 되어 있으므로 게임 서버에서는 매 로그인마다 이 API를 호출하지 않아야만 한다.

REQUEST

{“uid”:1234567, “amount”:123}

RESPONSE - SUCCESS

{“code”:0, “result”: { “paid”:123, “bonus”:23, “free”:3 }}

RESPONSE - FAIL

{“code”:-205, “result”:{“paid”:1, “bonus”:2,“free”:3}}

- result의 paid, bonus, free는 현재 잔액(int)이다.

ERROR

- -106 유저가 존재하지 않음. 조회 할 수 없으므로 잔액이 내려오지 않음
- -205 캐시 보유량이 부족함. 잔액이 함께 내려옴

/s/<gameid>/cash/use , /s/<gameid>/<channelnum>/cash/use

캐시를 사용한다. 만약 paid가 충분하다면 paid 부터 paid로 충분하지 않다면 bonus, free의 순서대로 사용된다. 사용하려는 양이 잔액을 초과하는 경우에는 에러이다. 위의 API에서 설명했듯이 지급 처리를 선행하고 호출하는 API인 만큼 사용이 실패하는 경우가 발생하지는 않는지 확인할 수 있어야 하며 문제 발생 시 파티게임즈와 협조하여 신속히 해결해야 한다.

item에는 캐시 사용 행위를 대표하는 문구를 detail에는 상세한 내용을 추가한다. 두 항목 모두 128자의 제한이 있으며 item은 반드시 포함되어야 하지만 detail은 포함하지 않아도 된다.

REQUEST

{"uid":1234567. "amount":123, "item":"Buy", "detail":"Object123"}

RESPONSE

```
{"code":0, "result":{
        "paid":0,
        "bonus":23,
        "free":3,
        "paid_diff":23,
        "bonus_diff":100,
        "free_diff":0
}}
```

- result의 paid, bonus, free는 현재 잔액(int)이다.
- paid_diff, bonus_diff, free_diff는 캐시 사용으로 인해 실제 차감된 paid, bonus, free의 양(int)이다.

ERROR

- -106 유저가 존재하지 않음
- -205 캐시 부족
- -206 캐시 사용량은 0보다 커야 한다.
- -903 item은 비어서는 안됨.

/s/<gameid>/cash/incfree , /s/<gameid>/<channelnum>/cash/incfree

유저에게 free캐시를 지급한다.

REQUEST

{"uid":1234567, "amount":123, "item":"lvup", "detail":1}

RESPONSE:

```
{
  "code":0,
  "result":
  {
        "paid":0,
        "bonus":23,
        "free":3,
        "free_diff":0
  }
}
```

- result의 paid, bonus, free는 현재 잔액(int)이다.
- free캐시만 지급되므로 paid, bonus는 제외하고 증가한 free_diff(int)만 내려온다.

ERROR

- -106 유저가 존재하지 않음
- -206 캐시 지급량은 0보다 커야 한다.

/s/<gameid>/cash/getlog , /s/<gameid>/<channelnum>/cash/getlog

유저의 캐시 충전 기록, 사용 기록 및 현재 잔액을 조회한다.

REQUEST

{“uid”:1234567}

RESPONSE

{"code": 0, "result": {

"paid":55, “bonus”:3, “free”:10,

“log”:[

{“date”:”2014-10-19 01:02:03”, “paid”:-1, …}

]

}}

- result의 paid, bonus, free는 현재 잔액(int)이다.
- result의 log는 충전 또는 사용 기록을 순서대로 나열한 배열이다.
- log 속의 paid, bonus, free는 증감값이다. (int)
- 사용 log는 item, detail 항목이 있고, 둘 다 스트링이다.
- 충전 log는 marketname, marketdata, productid, productname 항목이 있다.
- marketname = APPL, GOOG, TStore, NStore
- marketdata는 마켓별로 다른 의미의 스트링이다.
  - APPL = transaction id
  - GOOG = order id
  - TStore = txid 또는 플랫폼에서 발급한 sn이다.
  - NStore = paymentSeq

/s/<gameid>/gamelog/query

게임 로그를 얻어온다. 현재는 uid를 통해 조회하는 것만 제공된다. 고객 문의 처리시 사용하면 된다. 쿼리 시 반드시 기간을 지정해야 하고, 한 번에 31일 이상은 쿼리 할 수 없다. 실행이 매우 오래 걸릴 수도 있다.

REQUEST

{“startdate”: “2014-10-19”, “enddate”: “2014-10-19”, “uid”:1234567}

RESPONSE

```
{"code": 0, "result": {“log”: [
    {
        "time": "2014-09-02 08:46:53",
        "category": "Attend",
        "detail": {
            "d": 1
        }
    }
]}}
```

- 성공 시 result: result 속 log 배열에 로그들이 리턴된다.
- detail이 로그 상세 정보 json을 로딩한 것이다.

ERROR

- -903 날짜 형식 또는 기간에 문제가 있는 경우 리턴 될 수 있음

/s/<gameid>/promo/payment/daily, /s/<gameid>/<channelnum>/promo/payment/daily

월정액 프로모션의 지급을 요청한다. parameter에 월정액 프로모션의 sn을 포함해야 한다. (상품번호가 아닌 프로모션의 고유번호)

REQUEST

{"uid":1234567, "promosn":1}

RESPONSE:

```
{
  "code":0,
  "result":
  {
        "paid":0,
        "bonus":23,
        "free":3,
        "bonus_diff":0
        "gifts":"{}"
  }
}
```

- result의 paid, bonus, free는 현재 잔액(int)이다.
- bonus캐시만 지급되므로 paid, free는 제외하고 증가한 bonus_diff(int)만 내려온다.
- gifts에는 선물함으로 지급된 아이템 목록이 포함된다.

ERROR

- -609 gameid 혹은 promosn 값이 잘 못 됨.
- -610 월정액 프로모션에 해당하는 유저가 아님.
- -612 user id 혹은 channelnum가 잘 못 됨.

/s/<gameid>/unregister/list

게임에서 탈퇴하고, 탈퇴 철회기간이 끝나 최종 탈퇴된 유저 목록을 받아온다. 게임서버에서 주기적으로 호출하여 (하루 1회 이상 권장) 탈퇴된 유저의 데이터를 삭제해야 한다.

기간은 분 단위로 입력할 수 있고, 한 번 호출에 최대 2일간의 정보를 요청할 수 있다. 30일 지난 것은 받을 수 없다. 여러 호출끼리 기간이 중첩되어도 문제 없이 결과를 받을 수 있다.

REQUEST

{“starttime”: “2014-10-19 00:00”, “endtime”: “2014-10-19 06:00”}

- [starttime, endtime) : starttime은 포함하고, endtime은 포함하지 않는다.

RESPONSE

```
{"code": 0, "result":
    {
        "1234567": "2014-10-19 02:02:02”,
        “1234568”: “2014-10-19 02:02:03”,
    }
}
```

- 성공 시 result : uid와 최종 탈퇴 시각 매핑

ERROR

- -903 날짜 형식 또는 기간에 문제가 있는 경우 리턴 될 수 있음

/s/<gameid>/sendpush

특정 유저에게 푸시를 발송한다.

REQUEST

{“uid”: 12345678, “msg”: “푸시 메시지”}

RESPONSE

```
{"code": 0, "result":
    {
    }
}
```

ERROR

/s/<gameid>/dataversion/register

새로운 데이터 버전을 등록한다.

REQUEST

{"version": "1234_5678", "comment": "관리 페이지에서 보일 코멘트"}

RESPONSE

{"code": 0, "result": null}

ERROR

- -614 등록하려는 version이 이미 존재하면 발생
- -903 version이 비어있으면 발생

선물함 연동

모든 외부 채널과 통신한 결과가 선물함에 모이며, 개별 게임과는 선물함을 통해서 통신한다.

연동 작업의 큰 흐름은 다음과 같다.

1. 유저가 게임에 로그인 후, 선물함을 조회하는 시점에 SDK를 통해 선물 목록을 받아온다.
2. 게임 자체 선물함이 있는 경우, 목록을 합쳐서 여러 선물 종류들에 대해서 알맞게 클라이언트에 표시한다.
3. 클라이언트에서 선물 꺼내기 요청이 오면, 선물을 지급하고 선물 사용 API를 호출한다.

혹은, 선물함 목록을 받아와서 게임 자체 선물함에 맞게 변환해 넣는 방법이 있다. 이 경우에는 게임 자체 선물함에 지급 후 선물 사용 API를 호출한다.

선물함 사용을 위한 제반 사항

- 일반적인 종류: 캐시, 골드, 게임 아이템 등이 있으며 개발사에서 직접 백오피스에 등록/관리한다. (백오피스의 아이템 항목) 아이템의 구분을 위해 제공하는 필드는 t, n으로 t에는 문자열, n에는 숫자 값을 사용해야 하며, 일반적으로 각각 아이템 ID, 아이템 수량을 의미한다.
- 특수한 종류: 외부 게임 쿠폰 번호.

선물함 조회

선물함의 선물 목록 조회는 클라이언트에서 SDK를 이용한다. 자세한 내용은 SDK문서 선물함을 참고할 것.

/s/<gameid>/giftbox/use, /s/<gameid>/<channelnum>/giftbox/use

선물을 선물함에서 제거한다. 이 호출은 잘못 구현될 경우 여러 가지 문제가 발생할 수 있으니, 작성 시 많은 주의를 요한다.

우선 선물을 지급하고, API 호출을 나중에 하는 것이 권장된다. API 호출의 응답이 실패인 경우는 지급했던 선물을 회수하도록 한다. 데이터베이스 트랜잭션을 사용하면 간편하게 할 수 있다. API 호출 후 선물 지급을 하는 경우는 선물이 누락되어 고객 클레임이 발생할 수 있으니 권장하지 않는다.

주의해야 할 점은 호출이 connection은 성공했으나 timeout으로 실패한 경우다. 프로모션 시스템에서 선물은 제거되고 응답 패킷만 유실된 가능성이 있다. 이 경우 지급했던 것을 회수하면 선물이 누락된다. 따라서 API 호출은 정상적으로 완료되고, 실패 응답(code < 0)을 받았을 경우에만 지급했던 것을 회수하도록 한다.

게임 서버 로그에 선물 받았음을 sn과 함께 남길 것을 강력히 권장한다.

REQUEST

{"uid": 12345678901234567, "sn": 1}

- uid : 회원번호 (type: int)
- sn : 꺼낼 선물의 sn (type: int)

RESPONSE

- 실패 시 code
  - -10 : sn에 해당하는 선물이 없거나 다른 user의 선물
  - -11 : 이미 받은 선물
  - -999 : 그 외의 에러

프로모션 시스템

게임이 런칭하면 여러 가지 프로모션을 진행한다. 파티게임즈 내부 게임들과의 크로스 프로모션이나 외부 게임과도 크로스 프로모션을 하기도 하며, 게임 내적으로는 푸시를 보내 로그인을 유도하는 프로모션을 하기도 한다. 쿠폰을 발행하여 커뮤니티 이벤트를 벌여서 당첨자에게 선물을 지급하기도 한다. 이런 프로모션들을 다양하게 진행하되 개발사 부담을 최소로 하고자 프로모션 시스템을 제작하였다. 이 시스템을 이용하기 위해서는 반드시 파티게임즈에서 관리하는 선물함을 연동해야 한다.

/s/<gameid>/mission/complete

유저가 조건을 달성하였음을 프로모션 시스템에 알린다. 프로모션이 진행 중이면 보상이 선물함에 지급된다. 프로모션이 없을 때 호출해도 관계 없다.

중복 지급은 시스템에서 막혀 있으므로, 누락을 없애는 방향으로 신경을 써야 한다. 즉, 호출이 실패한 경우 나중에 다시 호출할 수 있도록 한다. 레벨 달성 이벤트 예를 들겠다. 레벨 달성 이벤트 발생시에 호출하여 실패할 수 있으므로, 호출 성공한 마지막 레벨을 저장해 놓는다. 추후 로그인이나 기타 다른 이벤트 발생시 현재 레벨과 마지막 호출 성공한 레벨을 비교하여 다르면 나중에 다시 호출할 수 있도록 한다.

REQUEST

{"uid": 12345678901234567, "mission": "reward1"}

- uid: 회원번호 (type: int)
- mission: 미리 협의된 미션 이름 (type: string)

RESPONSE

{"code": 0, "result": {"dup": 2, "new": 0}}

- 성공 시 result
  - new: 이번에 지급된 선물 수
  - dup: 기존에 지급된 선물 수

  현재 이 result는 별 의미가 없고, 조만간 수정될 예정이다. 현재는 new+dup이 미션에 걸려있는 프로모션 수라는 의미이나, 다른 게임에 지급된 경우도 합산되므로 큰 의미가 없다.

  추후에는 선물이 지급된 경우 메시지를 리턴하여 유저에게 알려줄 수 있게 하고, 선물함의 변화 여부를 알려줄 예정이다.

- 실패 시 code
  - -403 : 외부 게임 쿠폰 부족 (이 경우도 선물이 일부만 지급되진 않음)
  - -999 : 그 외의 에러

에러 목록

- -900 malformed request (잘못된 http 요청 또는 json 파싱 에러)
- -901 invalid hmac
- -902 missing parameter
- -903 invalid parameter

문서 이력

- 2014-10-19 0.1.0 최초 버전
- 2014-10-19 0.2.0 캐시 로그, 게임 로그 API 추가
- 2015-04-03 0.4.0 프로모션 관련 내용 통합