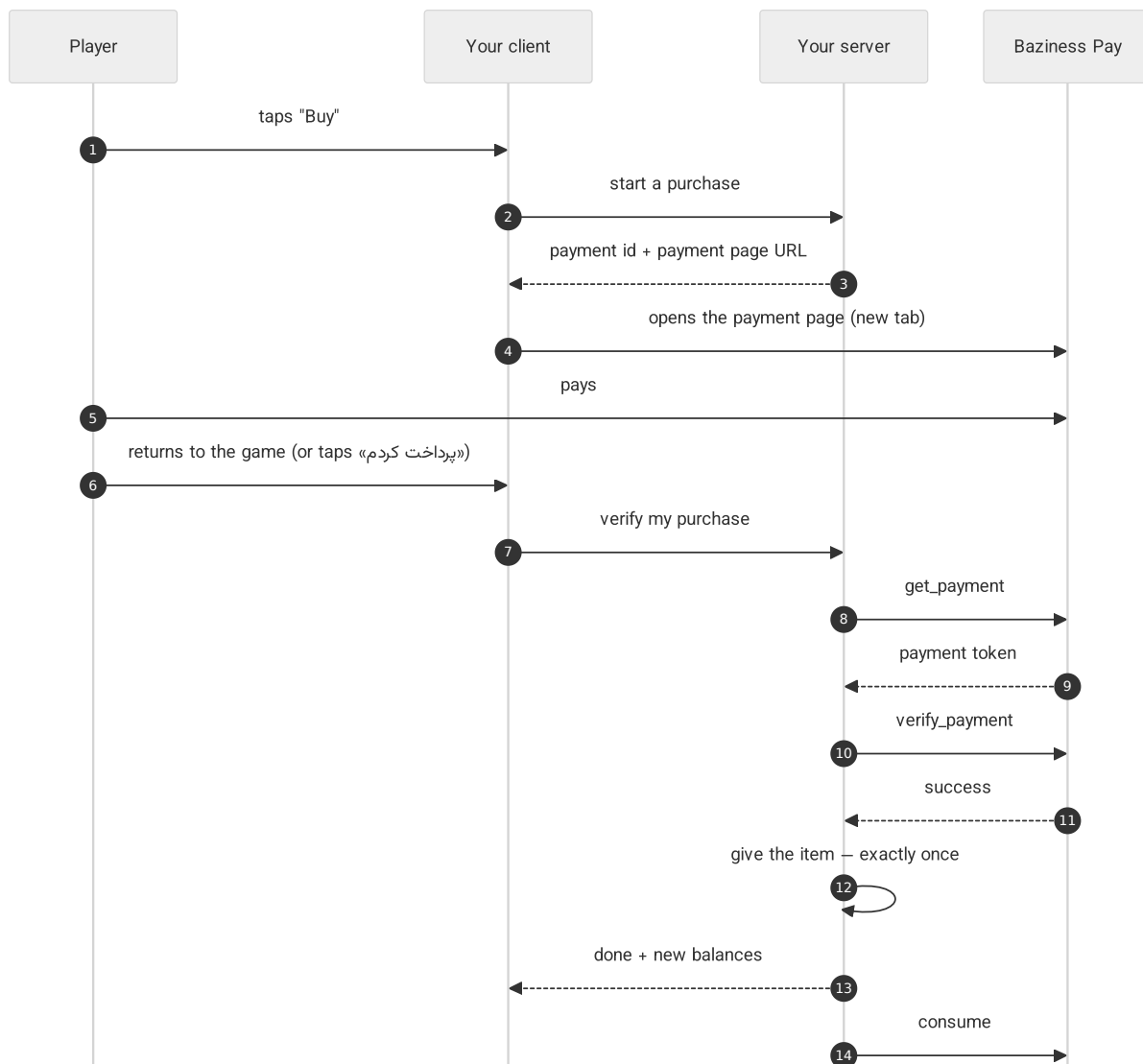


پرداخت بازی‌نس، قدم به قدم

فروش آیتم با پول واقعی، به ترتیب: صفحه‌ی پرداخت را باز کنید، پرداخت را پیدا و تأیید کنید، آیتم را دقیقاً یک بار بدهید و خرید را مصرف کنید تا دوباره قابل خرید باشد.

پیش از شروع

آدرس پایه {pay}	توسعه: <code>https://pay-dev.karizmastudios.org</code> اصلی: <code>https://pay.baziness.com</code>
چه کسی صدا می‌زند	سرور شما. کلاینت فقط صفحه‌ی پرداخت را باز می‌کند و بقیه‌ی کارها را با سرور خود شما انجام می‌دهد
pname	نام بسته‌ی پرداخت شما؛ ممکن است با نام بسته‌ی پلتفرم فرق کند
sku	شناسه‌ی محصولی که در داشبورد بازی‌نس ثبت شده (مثلاً <code>example_pack</code>)
uid	شناسه‌ی ثابت بازیکن در بازی شما، و باید عددی باشد (به شکل رشته بفرستید، مثلاً <code>"1042"</code>). uid غیرعددی با <code>no_required_params</code> رد می‌شود



قدم ۱ – باز کردن صفحه‌ی پرداخت

چه کار می‌کند: صفحه‌ی پرداخت بازینس را برای یک محصول به بازیکن نشان می‌دهد.

چطور: وقتی بازیکن «خرید» را می‌زند، کلاینت از **سرور شما** می‌خواهد خرید را شروع کند. سرور یک سفارش ذخیره می‌کند (قدم ۴ را ببینید)، آدرس صفحه را می‌سازد و برمی‌گرداند. کلاینت آن را در **تب جدید** (وب) یا مرورگر سیستم (موبایل) باز می‌کند.

HTTP

```
GET {pay}/buy?sku=example_pack&uid=1042&pname=com.karizmastudio.example
```

→ the hosted payment page (opened in a new tab / the system browser)

مرورگرها تبی را که **مستقیم** با ضربه‌ی بازیکن باز نشده باشد مسدود می‌کنند. اول در همان رویداد ضربه یک تب خالی باز کنید، بعد وقتی سرور جواب داد آن را به آدرس ببرید (**کد کلاینت**).

قدم ۲ – بررسی اینکه بازیکن پرداخت کرده یا نه

چه کار می‌کند: از بازیکنس می‌پرسد این بازیکن برای این محصول پرداخت کامل‌شده و هنوز مصرف‌نشده‌ای دارد یا نه، و اگر دارد توکن پرداخت را برمی‌گرداند.

کی صدایش بزنید: وقتی بازیکن به بازی برمی‌گردد (پنجره یا اپ دوباره فوکوس می‌گیرد)، وقتی «پرداخت کردم» را می‌زند، و وقتی فروشگاه باز می‌شود، برای پرداخت‌هایی که وقتی بازی بسته بود تمام شده‌اند.

HTTP

```
POST {pay}/get_payment
Content-Type: application/json

{ "uid": "1042", "sku": "example_pack", "pname": "com.karizmastudio.example" }
```

→ 200 OK (paid, not consumed yet)

```
{ "uid": "1042", "sku": "example_pack", "token": "TEST-633...", "time": 1791366000000,
  "success": true }
```

→ 200 OK (nothing to verify yet – the player hasn't paid)

```
{ "error": "payment_is_not_valid", "success": false }
```

یعنی «هنوز پرداختی نیست»: بی‌صدا منتظر بمانید. `payment_is_not_valid`

قدم ۳ – تأیید پرداخت

چه کار می‌کند: از بازیکنس تأیید می‌گیرد که توکن پرداخت واقعی است. این همان درخواستی است که از شما محافظت می‌کند: هیچ‌وقت بر اساس حرف کلاینت چیزی ندهید.

HTTP

```
POST {pay}/verify_payment
Content-Type: application/x-www-form-urlencoded

uid=1042&token=TEST-633...&sku=example_pack&pname=com.karizmastudio.example
```

→ 200 OK

```
{ "success": true }
```

`verify_payment` بدنه را به شکل فرم (`application/x-www-form-urlencoded`) می‌گیرد؛ بقیه‌ی درخواست‌ها JSON. فرستادن JSON در این‌جا رایج‌ترین دلیل شکست تأیید است.

قدم ۴ – دادن آیتم، دقیقاً یک بار

بازینس به شما می‌گوید پرداخت واقعی است؛ اینکه بازیکن آیتم را یک بار بگیرد کار سرور شماست. یک جدول پرداخت‌ها در دیتابیس نگه دارید:

معنی	ستون
شناسه‌ی سفارش شما (در قدم ۱ به کلاینت برمی‌گردد)	<code>id</code>
چه کسی چه چیزی می‌خرد	<code>user_id</code> ، <code>product_id</code> ، <code>sku</code> ، <code>price</code>
<code>REDIRECTED</code> (صفحه‌ی پرداخت باز شد) ← <code>FULFILLED</code> (آیتم داده شد)، یا <code>FAILED</code> / <code>CANCELLED</code>	<code>status</code>
دقیقاً همان چیزی که هنگام زدن «خرید» به بازیکن وعده داده شد	<code>reward_snapshot</code>
توکن پرداخت از قدم ۲، با محدودیت یکتا	<code>provider_token</code>

بعد از قدم ۳ موفق، در یک تراکنش دیتابیس:

۱. ردیف سفارش را قفل کنید و ببینید از قبل `FULFILLED` نباشد. اگر هست، بدون دادن دوباره‌ی آیتم «انجام شد» را برگردانید.
۲. اگر همین توکن پرداخت قبلاً سفارش دیگری را تحویل داده، این یکی را `FAILED` کنید و متوقف شوید.
۳. آیتم‌های `reward_snapshot` را بدهید، نه تعریف فعلی محصول را. مدیری که وسط پرداخت بازیکن یک بسته را ویرایش می‌کند نباید سفارش او را عوض کند.
۴. توکن را ذخیره کنید و سفارش را `FULFILLED` کنید.

قدم ۵ – مصرف کردن خرید

چه کار می‌کند: خرید را مصرف‌شده علامت می‌زند تا همان محصول دوباره قابل خرید باشد. تا وقتی مصرف نشده، قدم ۲ همچنان آن را برمی‌گرداند و خرید دوم آن SKU مسدود است.

کی: درست بعد از ثبت نهایی قدم ۴. اگر شکست خورد چند بار دوباره تلاش کنید؛ آیتم قبلاً داده شده، پس شکست این قدم هیچ وقت بازیکن را معطل نمی کند.

HTTP

POST {pay}/consume

Content-Type: application/json

```
{ "uid": "1042", "sku": "example_pack", "pname": "com.karizmastudio.example" }
```

→ 200 OK

```
{ "success": true }
```

چیزی که بازیکن می بیند

لحظه	کلاينت شما نشان می دهد
بعد از زدن «خرید»	یک پنجره ی کوچک: «صفحه ی پرداخت بازیکن باز شد، بعد از پرداخت به بازی برگرد»، با دکمه ی «پرداخت کردم»
برگشت به بازی، هنوز پرداختی پیدا نشده	چیز تازه ای نشان نمی دهد (بررسی بی صدا)؛ فقط زدن «پرداخت کردم» می گوید «پرداختی پیدا نشد، چند لحظه بعد دوباره بررسی کن»
پرداخت تأیید شد	«خرید موفق» و موجودی های به روز
پنجره پیش از پرداخت بسته شد	چیزی لغو نمی شود؛ دفعه ی بعد که فروشگاه باز شود دوباره بررسی می شود

تست

- روی **pay-dev** صفحه ی پرداخت گزینه ی «خرید آزمایشی» دارد: کل جریان، از قدم ۱ تا ۵، بدون پول واقعی اجرا می شود.
- اگر به هیچ درگاهی دسترسی ندارید، از **درگاه آزمایشی محلی** بازی نمونه استفاده کنید: همان پروتکل را دارد، با دکمه ی «پرداخت موفق» ([تنظیمات](#)).

بیلدهای فروشگاه (بازار / مایکت)

بیلدی که در کافه بازار یا مایکت منتشر می شود باید از **پرداخت درون برنامه ای خود آن فروشگاه** استفاده کند، نه پرداخت بازیکن. نام SKU ها را در هر دو یکی نگه دارید تا یک فهرست محصول برای همه ی بیلدها کافی باشد، و در آن بیلدها

`buildStore` را روی لینک صفحه‌های بازی‌نس بفرستید (پارامترها).