

API پلتفرم، قدم به قدم

همه‌ی endpointهای پلتفرم به ترتیبی که استفاده‌شان می‌کنید: چه کاری می‌کند، کی صدایش بزنید، چه بفرستید و چه برمی‌گردد. بدون کد؛ کد آماده در بخش «نمونه کد» است.

پیش از شروع

مهم

توکن‌های کاربر در حال تغییر به توکن دسترسی ۱۵ دقیقه‌ای، توکن تمدید روی سرور شما و کد اتصال یک‌بارمصرف هستند. قدم‌های زیر در دوره‌ی مهاجرت همچنان کار می‌کنند. چه چیزی را تا کی عوض کنید: **مهاجرت توکن**.

آدرس پایه {api}	توسعه: <code>https://platform-dev.karizmastudios.org/api</code> اصلی: <code>https://platform.karizmastudio.com/api</code>
قالب	بدنه‌ی درخواست و پاسخ JSON است (<code>Content-Type: application/json</code>) و نام فیلدها camelCase
چه کسی صدا می‌زند	سرور شما، به جز قدم ۶ (<code>user/get-coin</code>) که کلاینت بازیکن صدا می‌زند
احراز هویت	قدم‌های ۲ تا ۵ هدر <code><توکن کلاینت> Authorization: Bearer</code> از قدم ۱ را می‌خواهند؛ قدم ۶ توکن کاربر بازیکن را

کل ترتیب برای یک بازیکن جدید:



قدم ۱ – گرفتن توکن کلاینت

چه کار می‌کند: شناسه‌ی کلاینت و کلید محرمانه را با یک توکن کلاینت عوض می‌کند. همه‌ی درخواست‌های دیگر سرور (قدم‌های ۲ تا ۵) این توکن را در هدر **Authorization** می‌فرستند.

کی صدایش بزنید: با اولین درخواست به پلتفرم بعد از بالا آمدن سرور، و دوباره هر وقت درخواستی پاسخ **401** داد، چون توکن منقضی شده است.

HTTP

```

POST {api}/client/auth
Content-Type: application/json

{ "clientId": 12, "clientSecret": "....." }

→ 200 OK
{ "clientId": 12, "token": "eyJhbGciOiJIUzI1NiIs..." }
  
```


اگر `userId` برگشتی از قبل روی بازیکن دیگری از بازیکنان شما ذخیره شده، این همان شخص است، مثلاً روی گوشی جدید. آن را دوباره ذخیره نکنید؛ به جایش این دستگاه را به آن بازیکن جابه‌جا کنید (ورود ← جابه‌جایی حساب). روی ستونی که `userId` را نگه می‌دارد محدودیت یکتا بگذارید.

قدم ۳ – گرفتن توکن تازه برای کاربر

چه کار می‌کند: برای کاربر بازی‌نسی که از قبل می‌شناسید یک توکن تازه و مقدار فعلی `isLinkedToAccount` را برمی‌گرداند.

کی صدایش بزنید: در هر جلسه‌ی بعدی بازیکن، معمولاً هر بار که کلاینت بالا می‌آید. به این ترتیب کلاینت همیشه توکن معتبر دارد، و شما می‌فهمید کی یک کاربر ناشناس وارد حسابش شده (`isLinkedToAccount` شده `true`).

HTTP

```
GET {api}/client/get-user-token/239236
Authorization: Bearer <client token>

→ 200 OK
{ "userId": 239236, "token": "eyJhbGciOiJIUzI1NiIs...", "isLinkedToAccount": true }
```

قدم ۴ – بررسی توکنی که کلاینت فرستاده

چه کار می‌کند: می‌گوید یک توکن کاربر واقعی است یا نه، و مال کدام کاربر بازی‌نسی است.

کی صدایش بزنید: هر وقت توکنی از کلاینت رسید و می‌خواهید بر اساسش کاری بکنید. مورد اصلی: بازیکن در صفحه‌ی ورود بازی‌نسی وارد می‌شود، صفحه یک توکن به کلاینت می‌دهد و کلاینت آن را به سرور می‌فرستد. هیچ‌وقت بر اساس توکنی که بررسی نکرده‌اید حسابی وصل نکنید یا چیزی ندهید.

HTTP

```
GET {api}/client/validate-user-token/eyJhbGciOiJIUzI1NiIs...
Authorization: Bearer <client token>

→ 200 OK
{ "userId": 239236, "isValid": true }

→ 200 OK (an invalid or expired token is NOT an HTTP error)
{ "userId": 0, "isValid": false }
```

هشدار

توکن نامعتبر خطای HTTP نیست: پاسخ 200 با `"isValid": false` و `"userId": 0` است. همیشه `isValid` را بررسی کنید.

قدم ۵ – دادن گیتا کوین

چه کار می‌کند: به موجودی بازیکن گیتا کوین اضافه می‌کند و موجودی جدید را برمی‌گرداند.

کی صدایش بزنی: وقتی بازیکن در بازی شما سکه به دست می‌آورد؛ مثلاً با بردن یک مسابقه، رتبه‌ی تورنمنت یا باز کردن هدیه.

HTTP

```
POST {api}/client/add-coin
Authorization: Bearer <client token>
Content-Type: application/json

{
  "userId": 239236,
  "amount": 50,
  "transactionId": "239236_50_9f2c1a7d0b4e8c3f5a6d2e1b",
  "metaData": { "source": "match_win", "matchId": 17 }
}
```

→ 200 OK
{ "resourceId": 1, "amount": 2450 } ← the player's new balance

→ 400 Bad Request (this transactionId was already credited)
{ "success": false, "error": "add resource failed" }

فیلد	مقدار
userId	شناسه‌ی کاربر بازیکن از قدم ۲
amount	تعداد سکه (مثبت)
transactionId	یک شناسه‌ی یکتا برای همین پرداخت (پایین را ببینید). پلتفرم هر شناسه را فقط یک بار واریز می‌کند
metaData	هر JSON که می‌خواهید کنار تراکنش ذخیره شود (منبع، شناسه‌ی مسابقه و ...)، برای گزارش‌ها

امن کردن `transactionId`: آن را از ماهیت پرداخت بسازید، نه از زمان:

Text

```
transactionId = <userId>_<amount>_<first 24 hex chars of sha1(grantKey)>
grantKey       = "match:17" | "reward_claim:42" | "payment:88:GITA_COIN" | "tutorial_gift:
<userId>"
```

- یک پرداخت همیشه یک شناسه‌ی ثابت می‌سازد. تلاش دوباره، دو بار زدن دکمه یا درخواست تکراری با **400** رد می‌شود و هیچ‌وقت دو بار پرداخت نمی‌شود.
- این **400** را به معنی «قبلاً پرداخت شده» بگیرید و حسابداری خودتان را تمام کنید، مثلاً جایزه را دریافت شده علامت بزنید. وگرنه کرسی بین واریز و ذخیره‌ی شما بازیکن را پشت یک خطا گیر می‌اندازد.
- شناسه‌ای که از ساعت ساخته شود (<userId>_<amount>_<time>) با هر بار صدا زدن دوباره واریز می‌کند؛ فقط برای پرداخت‌هایی که واقعاً تکرارشونده‌اند.

قدم ۶ – خواندن موجودی (کلاینت)

چه کار می‌کند: موجودی گیتا کوین بازیکن را برمی‌گرداند. تنها درخواست پلتفرم است که **کلاینت** با **توکن کاربر** بازیکن می‌زند. درخواست از دامنه‌های دیگر مجاز است، پس بازی مرورگری می‌تواند مستقیم صدایش بزند.

کی صدایش بزنید: بعد از اینکه کلاینت توکن گرفت، وقتی یک صفحه‌ی بازیکن بسته می‌شود (شاید بازیکن سکه خرج کرده باشد)، وقتی بازی دوباره به جلو می‌آید، و بعد از اینکه سرور سکه داد.

HTTP

```
GET {api}/user/get-coin
Authorization: Bearer <user token>

→ 200 OK
{ "resourceId": 1, "amount": 2450 }
```

خطاها در یک نگاه

پاسخ	کجا	معنی	کار شما
401 Unauthorized	قدم‌های ۲ تا ۵	توکن کلاینت نیست یا منقضی شده	قدم ۱ را دوباره صدا بزنید و یک بار دیگر بفرستید
200 با "isValid": false	قدم ۴	توکن واقعی نیست یا منقضی شده	درخواست را رد کنید؛ کلاینت باید دوباره وارد شود
400 با "add resource failed"	قدم ۵	این transactionId قبلاً واریز شده	آن را پرداخت‌شده حساب کنید
خطای شبکه / 5xx	هر جا	پلتفرم در دسترس نیست	بازی را بدون بخش‌های بازی‌نس ادامه دهید و بعداً دوباره تلاش کنید