

Baziness Developer Wiki

Integrate Baziness login, Gita coins, reward pages and Baziness Pay into your game.

1. [Introduction](#)
2. [Getting started, step by step](#)
3. [Platform API, step by step](#)
4. [Baziness pages & their messages](#)
5. [Baziness Pay, step by step](#)
6. [Login & connecting a Baziness account](#)
7. [Gita coins](#)
8. [Reward page \(«انتخاب جایزه»\)](#)
9. [Tutorial offer page \(game-offer\)](#)
10. [Links & UTM parameters](#)
11. [Migrating to link codes & short-lived tokens](#)
12. [Server code](#)
13. [Client code](#)
14. [Bottom sheet \(in-game web view\)](#)
15. [Example game — Rock, Paper, Scissors](#)
16. [Go-live checklist & troubleshooting](#)
17. [API reference](#)

Introduction

What Baziness gives your game, how the parts fit together, and how this wiki is organised.

What Baziness is

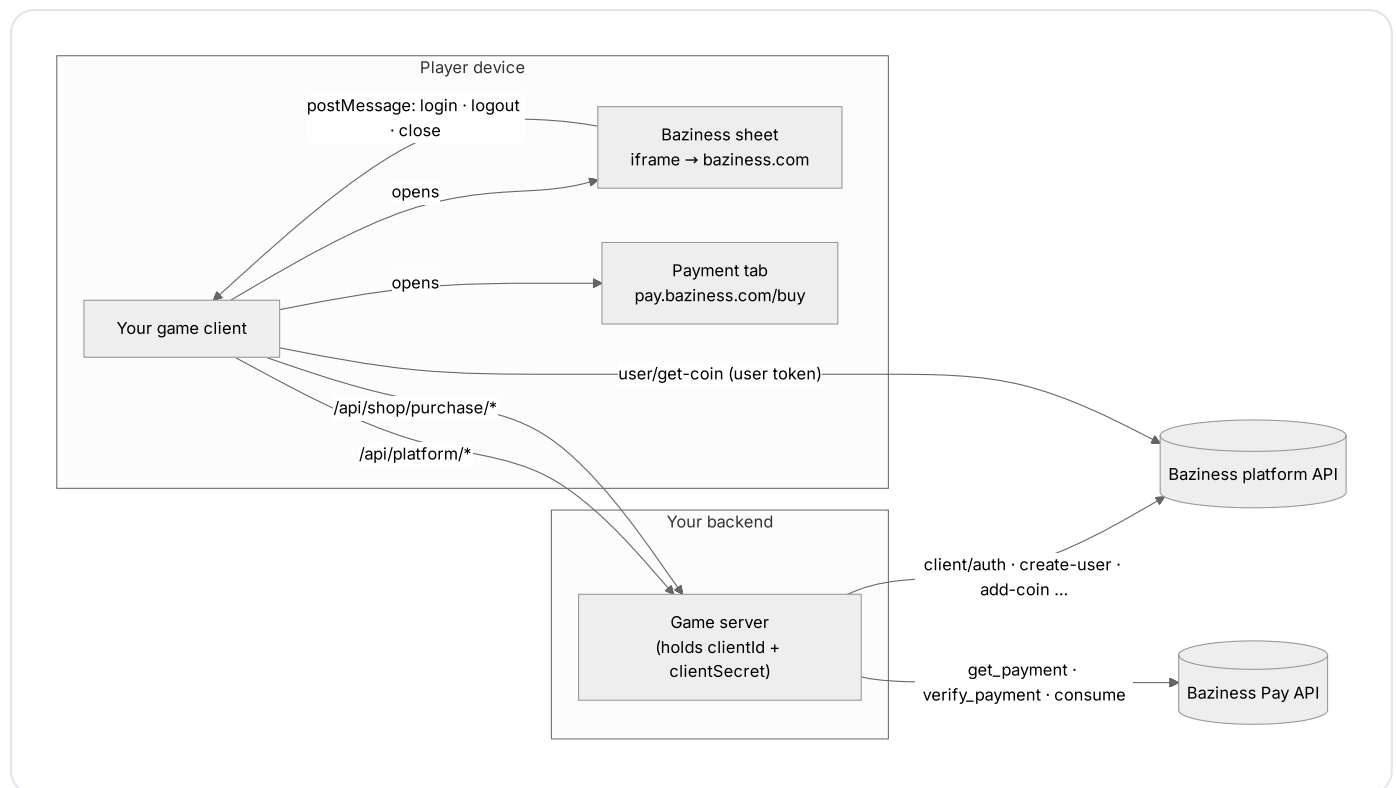
Baziness is the game platform of **Gita Games**. When your game integrates it, your players get:

- **A Baziness account:** they sign in with their phone number, and the same account works in every Baziness game.
- **Gita coins** (گیتا کوین): the platform currency. Players earn it in games and spend it on real prizes.
- **Baziness pages inside your game:** sign-in, profile, the «انتخاب جایزه» reward shop, and your game's own offer page.
- **Baziness Pay:** real-money purchases through a hosted payment page.

These come as two **separate** integrations. You can ship either one alone, or both together:

Integration	What the player gets	Who talks to Baziness	What you need
Baziness platform	accounts, Gita coins, Baziness pages	your server calls the platform API; your client shows the pages and reads the coin balance	client id, client secret, platform package name
Baziness Pay	real-money purchases	your server calls the Pay API; your client opens the payment page	Pay package name, SKUs registered in the Baziness dashboard

How the parts fit together



The client calls Baziness directly in only two cases: it shows Baziness **pages**, and it reads the coin balance (`user/get-coin`) with the player's own token. Everything else goes through **your server**, because only your server holds the client secret.

The five golden rules

1. **The client secret never leaves your server.** The browser or app gets only public settings and the player's own token.
2. **The server decides and the client only displays.** Account links, coin grants and purchases are always checked on the server.
3. **Every Baziness page link carries `packageName` and `sdkType=js`.** Without `sdkType=js` the page can't talk to your game, so it can't even close itself.
4. **The game keeps working without Baziness.** If the platform is switched off or unreachable, hide the Baziness parts and carry on.
5. **Every payout has a unique, repeatable id.** A retry, a double tap or a crash must never pay twice.

Environments

	Development (test games)	Production
Platform API	https://platform-dev.karizmastudios.org/api	https://platform.karizmastudio.com/api
Baziness pages	https://platform-web-dev.karizmastudios.org	https://baziness.com
Baziness Pay	https://pay-dev.karizmastudios.org	https://pay.baziness.com
Signing in	any phone number, code 111111 (no SMS is sent)	real phone number and SMS code
Test purchases	«خرید آزمایشی» on the payment page (no real payment)	real payment

Each environment has **its own client id and secret**.

How this wiki is organised

Section	What is in it	Code?
Getting started	this page, and the integration step by step	no
API	every endpoint in order: what it does, when to call it, request, response and rules	only requests and responses
Guides	the flows: login, Gita coins, the reward page, the tutorial offer page, links	no
Code samples	ready-made server code (TypeScript, C#, Python) and client code (web, React, Cocos Creator, Unity), including the web view	yes
Example game	a complete open-source game that uses everything here: play it and download it	the whole project
Reference	the go-live checklist, troubleshooting, and every endpoint in one table	no

Every page can be downloaded as **Markdown** or **PDF** from the buttons under its title. The sidebar has the full guide in both formats.

Getting started, step by step

What to ask the platform team for, where each value goes, and the whole integration as a short list of steps, each linking to its details.

1. Ask for your credentials

The Baziness platform team gives you these values, **one set per environment** (development and production):

Value	Example	Used by	Secret?
Client id	12	your server, to get a client token	no, but keep it private
Client secret		your server, to get a client token	yes — server only
Platform package name	com.karizmastudio.example	every Baziness page link (<code>packageName</code>)	no
Game name	example	the reward page and the game offer page (<code>game_name</code>)	no
Pay package name	com.karizmastudio.example	every Baziness Pay call (<code>pname</code>); may differ from the platform package	no
SKUs	example_pack	the products you sell with Baziness Pay	no

Warning

The client secret must **never** reach a browser or an app build: not in client code, not in a `VITE_*` or `NEXT_PUBLIC_*` variable, not in git. Anyone holding it can create players and **give Gita coins** as your game.

2. Understand the two tokens

Everything on the platform uses one of two tokens. Keeping them apart is the key to the whole integration:

Token	Who has it	How it is obtained	What it is for
Client token	your server only	client id + client secret → <code>client/auth</code>	every server-to-server call: creating players, checking tokens, giving coins
User token	the player's client (from your server)	<code>client/create-user</code> or <code>client/get-user-token</code>	opening Baziness pages as this player (<code>linkedToken</code>), reading their coin balance

3. Follow the steps

#	Step	Where	Details
1	Your server gets a client token with the client id and secret	server	Platform API → step 1
2	On a player's first visit, your server creates their Baziness user and saves its id. On later visits it gets a fresh user token	server	steps 2–3
3	Your client shows Baziness pages (sign-in, profile, rewards) with the user token, and listens to their messages	client	Baziness pages
4	When the player signs in on a page, your server checks the token and links the account (or switches to the player's other account)	server	Login, step 4
5	Your client shows the Gita coin balance , and your server gives Gita coins for wins and rewards	both	Gita coins, steps 5–6
6	The «انتخاب جايزه» button opens the reward page ; the tutorial can open your game offer page	client	Reward page, Tutorial offer page
7	Sell items with Baziness Pay	both	Baziness Pay
8	Tag the links that bring players in with UTM parameters	both	Links & UTM

Ready-made code for every step is in [Server code](#) and [Client code](#). The [example game](#) puts it all together.

4. Configure your server

The example game and the code samples read these settings. Use the same names, so samples can be copied unchanged:

Setting	Example	Meaning
BAZINESS_PLATFORM_ENABLED	true	master switch. While false, the game shows no Baziness UI at all
BAZINESS_PLATFORM_USE_PRODUCTION	false	false = development URLs, true = production URLs
BAZINESS_CLIENT_ID	12	client id
BAZINESS_CLIENT_SECRET		client secret (secret)
BAZINESS_PLATFORM_PACKAGE_NAME	com.karizmastudio.example	packageName on page links
BAZINESS_GAME_NAME	example	game_name on the reward and offer pages
BAZINESS_PAY_ENABLED	true	turns the buy buttons on
BAZINESS_PAY_BASE_URL	https://pay-dev.karizmastudios.org	the Pay server for this environment
BAZINESS_PAY_PACKAGE_NAME	com.karizmastudio.example	pname on every Pay call
BAZINESS_SKU_PREFIX	(empty)	optional: SKU = prefix + product id

The platform counts as **on** only when `BAZINESS_PLATFORM_ENABLED=true` **and** both the client id and the secret are set. A half-configured server then behaves as if Baziness didn't exist.

Platform API, step by step

Every platform endpoint in the order you use it — what it does, when to call it, what to send and what comes back. No code; ready-made code is in the Code samples section.

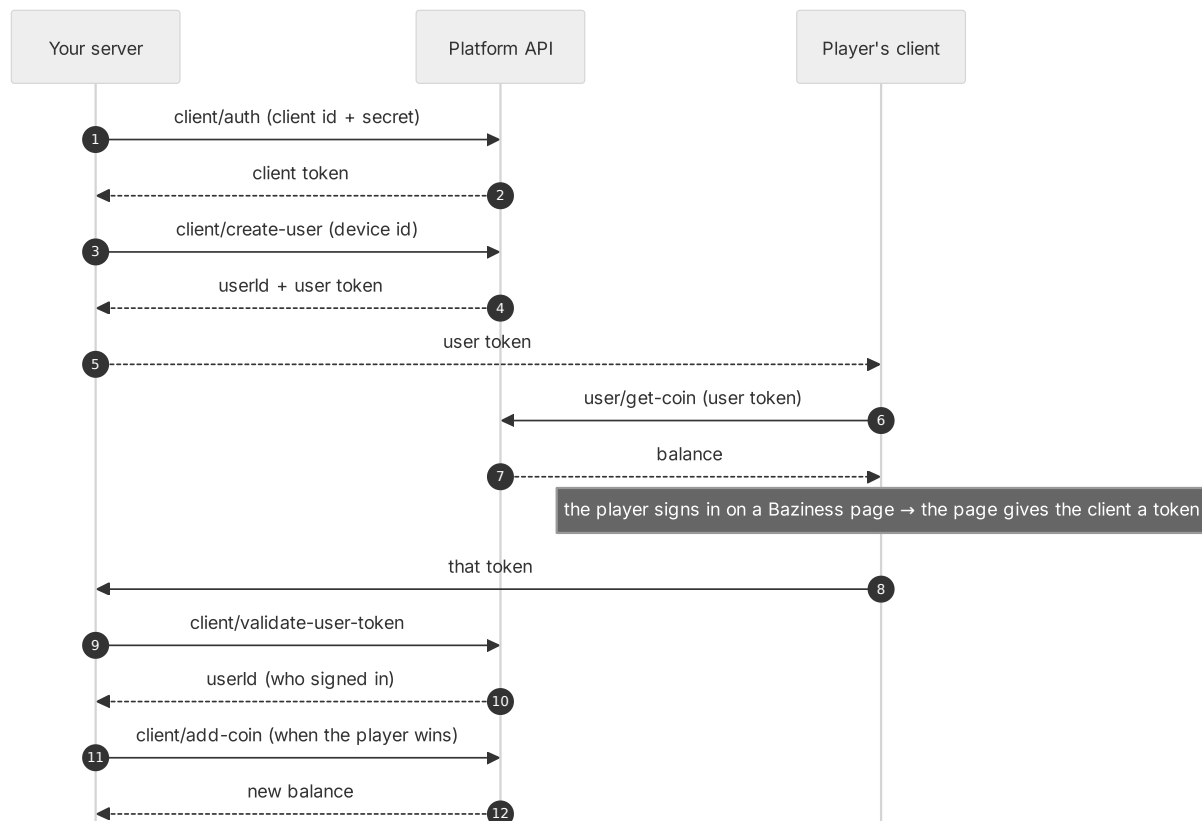
Before you start

Important

User tokens are moving to **15-minute access tokens, refresh tokens kept on your server, and one-time link codes**. The steps below still work during the migration. What to change, and by when: [Token migration](#).

Base URL	development: <code>https://platform-dev.karizmastudios.org/api</code> {api} production: <code>https://platform.karizmastudio.com/api</code>
Format	JSON request and response bodies (<code>Content-Type: application/json</code>), field names in camelCase
Who calls it	your server , except step 6 (<code>user/get-coin</code>), which the player's client calls
Authentication	steps 2–5 need <code>Authorization: Bearer <client token></code> from step 1; step 6 needs the player's user token

The whole sequence for a new player:



Step 1 — Get a client token

What it does: exchanges your **client id** and **client secret** for a **client token**. Every other server call (steps 2–5) sends this token in the **Authorization** header.

When to call it: on the first platform call after your server starts, and again whenever any call answers **401**, because the token has expired.

HTTP

```

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

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

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

Rules

- Keep the token **in your server's memory** and reuse it for every call. Don't request a new one per call.
- When a call answers **401 Unauthorized**, call **client/auth** again and **retry that call once**.
- The client token is as sensitive as the secret: never send it to a client.

Step 2 — Create a Baziness user for a player

What it does: gives one of your players a Baziness user. It returns:

- the Baziness `userId` — save it on your player's record; you need it for every later call
- a **user token** for this player — send it to your client
- `isLinkedToAccount` — whether this user has signed in with a phone number (`true`) or is still anonymous (`false`)

When to call it: the first time a player of yours needs Baziness, which is usually their first launch while the platform is on. If you already have a `userId` saved for the player, use step 3 instead.

HTTP

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

{ "deviceId": "8f0c2b7e-...", "platform": "Web", "store": "Direct", "linkedToken": null }
```

→ 200 OK

```
{ "userId": 239236, "token": "eyJhbGciOiJIUzI1NiIs... ", "isLinkedToAccount": false }
```

Field	Value
<code>deviceId</code>	your player's stable id, for example the device id your game generated on first launch
<code>platform</code>	<code>Web</code> , <code>Android</code> , <code>Ios</code> or <code>Editor</code>
<code>store</code>	<code>Direct</code> (web), <code>GooglePlay</code> , <code>AppStore</code> , <code>CafeBazaar</code> or <code>Myket</code>
<code>linkedToken</code>	a user token the player already has, or <code>null</code> . The Baziness app opens games with <code>?linkedToken=...</code> ; passing it here binds the player to that Baziness account

Important

If the returned `userId` is **already saved on another of your players**, it is the same person, for example on a new phone. Don't save it a second time. Switch this device to that player instead (see [Login → account switching](#)). Put a **unique** constraint on the column that stores the `userId` .

Step 3 — Get a fresh user token

What it does: returns a new user token for a Baziness user you already know, and their current `isLinkedToAccount`.

When to call it: on every later session of the player, typically each time your client boots. Then the client always has a valid token, and you learn when an anonymous user has signed in (`isLinkedToAccount` became `true`).

HTTP

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

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

Step 4 — Verify a token from the client

What it does: tells you whether a user token is genuine, and **which Baziness user it belongs to**.

When to call it: whenever a token arrives from a client and you're about to act on it. The main case: the player signs in on the Baziness login page, the page gives your client a token, and your client sends it to your server. **Never link an account or give anything based on a token you haven't verified.**

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 }
```

Warning

An invalid token is **not** an HTTP error: the answer is `200` with `"isValid": false` and `"userId": 0`. Always check `isValid`.

Step 5 — Give Gita coins

What it does: adds Gita coins to a player's Baziness balance and returns the new balance.

When to call it: when the player earns coins in your game, for example by winning a match, ranking in a tournament or opening a gift.

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" }

Field	Value
<code>userId</code>	the Baziness user id from step 2
<code>amount</code>	how many coins to add (positive)
<code>transactionId</code>	a unique id for this payout (see below). The platform credits each id only once
<code>metaData</code>	any JSON you want stored with the transaction (source, match id ...), useful for reports

Making `transactionId` safe: build it from **what the payout is**, not from the time:

Text

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

- The same payout always produces the same id. A retry, a double tap or a repeated request is refused with `400` and never pays twice.
- **Treat that `400` as "already paid"** and finish your own bookkeeping, for example marking the reward as claimed. Otherwise a crash between the credit and your save leaves the player stuck on an error.

- An id made from the clock (`<userId>_<amount>_<time>`) credits again on every call. Use it only for grants that are genuinely repeatable.

Step 6 — Read the balance (client)

What it does: returns the player's Gita coin balance. This is the only platform call the **client** makes, with the player's **user token**. Cross-origin requests are allowed, so a browser game can call it directly.

When to call it: after the client receives the token, when a Baziness page closes (the player may have spent coins), when the game comes back to the foreground, and after your server gave coins.

HTTP

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

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

Errors at a glance

Answer	Where	Meaning	What to do
<code>401 Unauthorized</code>	steps 2–5	missing or expired client token	call step 1 again, retry once
<code>200</code> + <code>"isValid": false</code>	step 4	the token is not genuine or has expired	refuse the request; the client should sign in again
<code>400</code> + <code>"add resource failed"</code>	step 5	this <code>transactionId</code> was already credited	treat it as paid
network error / <code>5xx</code>	any	the platform is unreachable	keep the game running without Baziness features; retry later

Baziness pages & their messages

The web pages your game shows inside itself — sign-in, profile, the reward shop and your game's offer page — the parameters each one needs, and the messages they send back.

What the pages are

Some parts of Baziness are **web pages** that your game shows **inside itself**, for example the sign-in screen and the «انتخاب جایزه» reward shop. You open a page's URL in an in-game web view: an `iframe` on the web, a `WebView` component in an engine. The page talks back to your game with **messages**, for example "the player signed in" or "close me".

All the pages are hosted on one site:

Environment	Pages base {web}
Development	<code>https://platform-web-dev.karizmastudios.org</code>
Production	<code>https://baziness.com</code>

How to build the in-game view (a sheet that slides up and is swiped down to close) is in [Bottom sheet / web view](#).

Parameters

Important

`linkedToken` is being replaced by `linkCode`, a one-time code from your server (`v2/client/link-code/{userId}`). Pages accept both during the migration. See [Token migration](#).

Parameter	When	Value	Why
<code>packageName</code>	always	your platform package name	identifies your game
<code>sdkType</code>	always	<code>js</code>	tells the page it is inside a game, so it sends you messages. Without it, even the page's close button can't close your view
<code>linkedToken</code>	whenever you have one	the player's user token	the page acts as this player, and a sign-in on the page is bound to them
<code>redirect</code>	profile, rewards	<code>profile</code> / <code>shop</code>	where the page returns after a sign-in inside it
<code>allowGuest</code>	sign-in, profile, game offer	<code>false</code>	hides the page's own "continue as guest"; your game already offers that choice
<code>game_name</code>	rewards, game offer	your game name, URL-encoded	selects your game's prizes and offer
<code>buildStore</code>	store builds only	<code>CafeBazaar</code> / <code>Myket</code>	the page hides links to the other stores

Encode every value, because `game_name` may be Persian and tokens may contain `+` or `/`.

The pages

Page	Path	Open it when	Parameters
Sign-in	/	the player taps «ورود با» حساب بازینس, or must sign in to claim a reward	packageName , sdkType=js , linkedToken , allowGuest=false
Profile	/profile	Settings → connect or manage the account	packageName , sdkType=js , linkedToken , redirect=profile , allowGuest=false
Reward shop «انتخاب جایزه»	/shop	the player taps your reward button	game_name , packageName , sdkType=js , linkedToken , redirect=shop . No allowGuest
Game offer	/game-offer	once, during your tutorial (details)	game_name , packageName , sdkType=js , linkedToken , allowGuest=false

Text

```
Login      https://baziness.com/?
packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...&allowGuest=false
Profile    https://baziness.com/profile?
packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...
&redirect=profile&allowGuest=false
Rewards    https://baziness.com/shop?
game_name=example&packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...
&redirect=shop
Game offer https://baziness.com/game-offer?
game_name=example&packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...
&allowGuest=false
```

Important

For the reward shop, open `/shop` . Don't open the site root (`/`): the root sends a player inside a game to the sign-in screen, while `/shop` uses the token the player already has.

Messages from the pages

A page sends messages with `postMessage` . Each message is JSON, as a string or an object, in the form `{ "type": "...", "data": ... }` :

type	data	Meaning	What your game does
close	—	the page's close button was pressed	close the view
login	a token: " <code><token></code> " or <code>{ "token": "<code><token></code>" }</code>	the player signed in to an account	send the token to your server , which verifies it and links the account or switches to the player's other account; then refresh the token and close the view
login	(none)	the player's current account was linked in place	mark the player as linked, refresh the token
token	a token	a silent handshake the page sends when it loads	ignore it: it is not a sign-in
logout	—	the player signed out inside the page	close the view, delete everything your game stored locally, and start again as a brand-new visitor

Warning

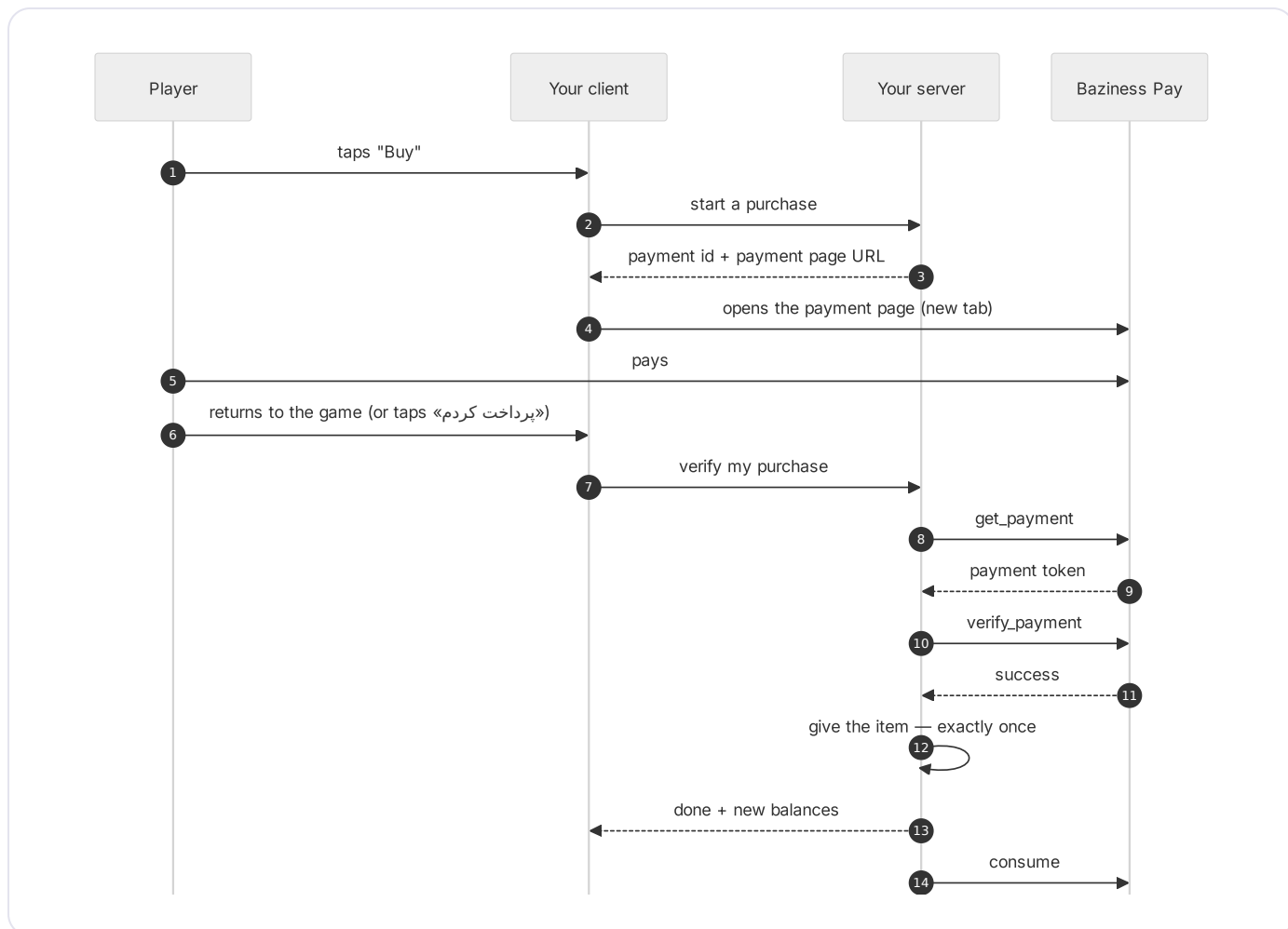
- Only accept messages whose origin is the page you opened (`{web}`), and ignore everything else.
- Never treat `token` as a sign-in. If you do, the reward shop closes as soon as it opens.

Baziness Pay, step by step

Selling items for real money, in order — open the payment page, check and verify the payment, give the item exactly once, and consume the purchase so it can be bought again.

Before you start

Base URL <code>{pay}</code>	development: <code>https://pay-dev.karizmastudios.org</code> production: <code>https://pay.baziness.com</code>
Who calls it	your server. The client only opens the payment page, and otherwise calls your own server
<code>pname</code>	your Pay package name . It may differ from the platform package name
<code>sku</code>	the product id registered in the Baziness dashboard (for example <code>example_pack</code>)
<code>uid</code>	your stable id for the player, and it must be numeric (send it as a string, e.g. <code>"1042"</code>). A non-numeric uid is rejected with <code>no_required_params</code>



Step 1 — Open the payment page

What it does: shows the player Baziness's payment page for one product.

How: when the player taps "Buy", your client asks **your server** to start a purchase. Your server saves an order (see step 4), builds the page URL and returns it. The client opens it in a **new tab** on the web, or in the system browser on mobile.

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)

Tip

Browsers block new tabs that aren't opened **directly** by a tap. Open an empty tab in the tap handler first, then send it to the URL once your server answers ([client code](#)).

Step 2 — Check whether the player paid

What it does: asks Baziness whether this player has a completed, not yet consumed payment for this product, and if so returns its **payment token**.

When to call it: when the player comes back to the game (the window or app regains focus), when they tap «پرداخت کردم» ("I've paid"), and when your shop opens, to catch payments finished while the game was closed.

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 }
```

`payment_is_not_valid` simply means "nothing paid yet": keep waiting quietly.

Step 3 — Verify the payment

What it does: confirms with Baziness that the payment token is genuine. **This is the call that protects you:** never give anything based on what the client says.

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 }
```

Warning

`verify_payment` takes a **form** body (`application/x-www-form-urlencoded`). The other calls take JSON. Sending JSON here is the most common reason verification fails.

Step 4 — Give the item exactly once

Baziness tells you the payment is real. Making sure the player receives the item **once** is your server's job. Keep a **payments table** in your database:

Column	Meaning
<code>id</code>	your order id (returned to the client in step 1)
<code>user_id</code> , <code>product_id</code> , <code>sku</code> , <code>price</code>	who buys what
<code>status</code>	<code>REDIRECTED</code> (payment page opened) → <code>FULFILLED</code> (item given), or <code>FAILED</code> / <code>CANCELLED</code>
<code>reward_snapshot</code>	exactly what the player was promised when they tapped "Buy"
<code>provider_token</code>	the payment token from step 2, with a unique constraint

Then, after a successful step 3, in **one database transaction**:

1. Lock the order row and check it isn't already `FULFILLED` . If it is, answer "done" without giving anything again.
2. If the same payment token already fulfilled **another** order, mark this one `FAILED` and stop.
3. Give the items in `reward_snapshot` , not the product's current definition. An admin who edits a pack while the player pays must not change the order.
4. Save the token and set the order to `FULFILLED` .

Step 5 — Consume the purchase

What it does: marks the purchase as used, so the same product can be bought again. Until it is consumed, step 2 keeps returning it and a second purchase of the SKU is blocked.

When: right **after** step 4 is committed. Try a few times if it fails. The item is already given, so a failure here never blocks the player.

HTTP

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

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

```
→ 200 OK
{ "success": true }
```

What the player sees

Moment	Your client shows
after tapping "Buy"	a small window: "the Baziness payment page is open, come back after paying", with a «پرداخت کردم» button
back in the game, payment not found yet	nothing new (quiet re-check); only a tap on «پرداخت کردم» says "payment not found yet, try again in a moment"
payment verified	"purchase successful", and the updated balances
window closed before paying	nothing is cancelled; the payment is checked again the next time the shop opens

Testing

- On **pay-dev**, the payment page offers «خرید آزمایشی» (test purchase): the whole flow, from step 1 to step 5, runs without real money.
- Without any gateway access, use the **local fake gateway** from the example game: it speaks the same protocol, with a «پرداخت موفق» button ([settings](#)).

Store builds (Cafe Bazaar / Myket)

A build published in Cafe Bazaar or Myket must use **that store's own in-app billing** instead of Baziness Pay. Keep the same SKU names in both, so one product list serves every build, and send `buildStore` on Baziness page links in those builds ([parameters](#)).

Login & connecting a Baziness account

How a player gets a Baziness identity, signs in from the splash screen or Settings, and how your game switches accounts safely.

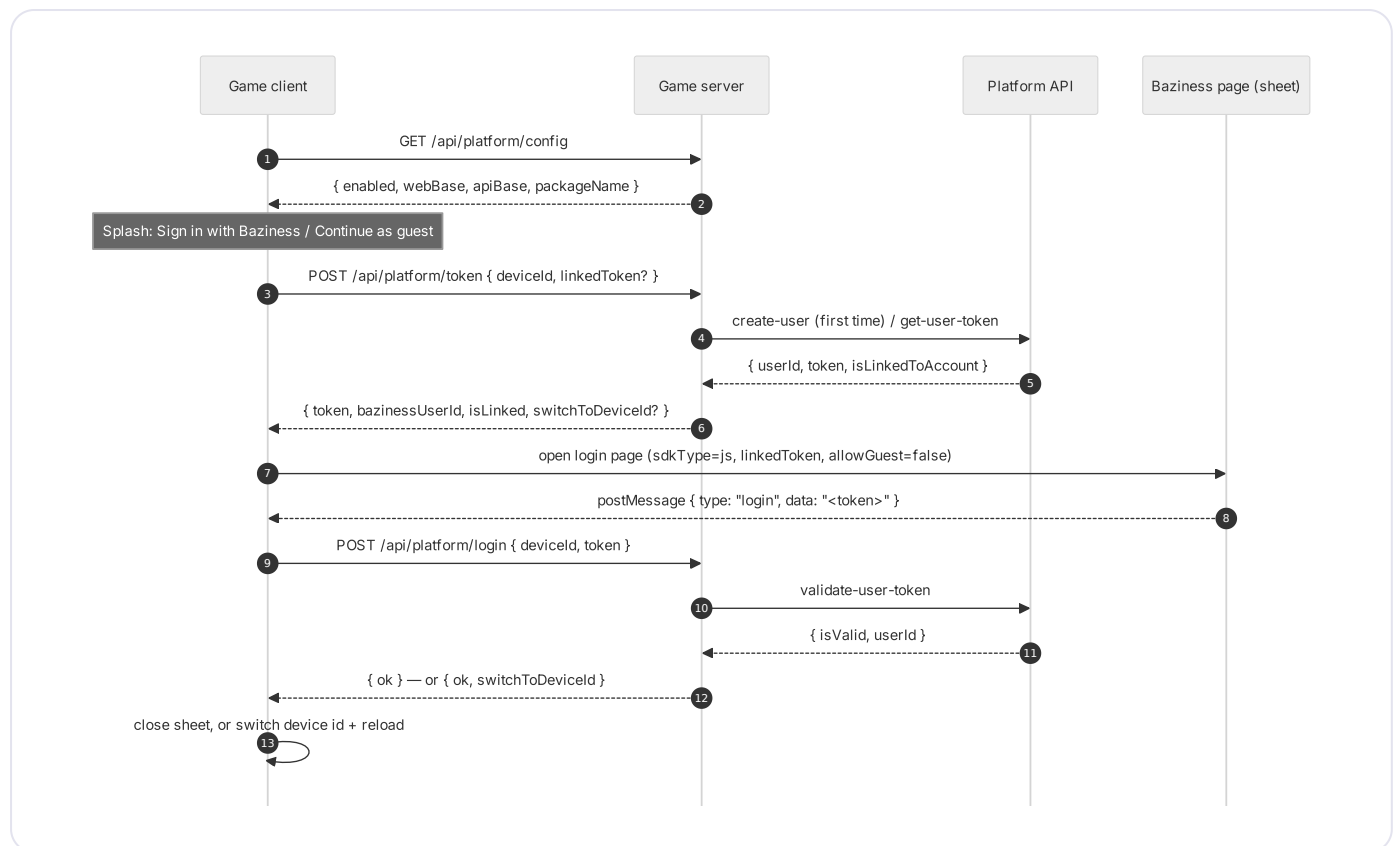
The identity model

Your game keeps **its own player accounts**, for example keyed by a device id generated on first launch. Baziness is attached **on top**:

- 1. The first time the platform is on, your server creates an **anonymous Baziness user** for the player (`create-user`) and saves its `userId` on the player's record.
- 2. The player stays **anonymous** (`isLinkedToAccount: false`) until they sign in with their phone number on a Baziness page. The account is then **linked**.
- 3. If the Baziness account they sign into **already belongs to another of your players** (the same person on another phone), your game **switches** to that player. It doesn't link the account twice.

State	Baziness <code>userId</code> saved?	<code>isLinkedToAccount</code>	What the player sees
Baziness off	—	—	no Baziness UI
Guest	yes (anonymous user)	<code>false</code>	coin balance and reward button work; "connect your account" in Settings
Signed in	yes	<code>true</code>	everything, including claiming Gita coin rewards

The whole flow



What your server provides

Your client never calls the platform's server API itself. It calls **three endpoints of your own server**:

Your endpoint	The client sends	Your server does	It answers
GET <code>/api/platform/config</code>	—	nothing; it only reports settings	<code>{ enabled, webBase, apiBase, packageName, gameName, gameOfferUrl }</code> , or <code>{ enabled: false }</code>
POST <code>/api/platform/token</code>	device id, <code>linkedToken</code> if any	no Baziness user yet → <code>create-user</code> ; otherwise → <code>get-user-token</code> . If the resulting <code>userId</code> belongs to another player, it does not link	<code>{ token, userId, isLinkedToAccount }</code> , or <code>{ switchToDeviceId }</code>
POST <code>/api/platform/login</code>	device id, the token from the sign-in page	<code>validate-user-token</code> → whose account is it? Another player's → switch; otherwise → link to this player	<code>{ ok: true }</code> , or <code>{ ok: true, switchToDeviceId }</code> , or <code>{ ok: false }</code>

If the platform is unreachable, `/token` answers `{ enabled: true, error: "platform_unavailable" }` without a token, and the game boots anyway.

The first launch: guest or sign in

When Baziness is on and the player has neither chosen yet nor signed in, the splash screen shows two buttons instead of the loading bar:

- «ورود با حساب بازینس» (sign in with Baziness): opens the sign-in page and waits.
- «ادامه به عنوان مهمان» (continue as guest): remembers the choice and continues.

The boot order:

1. Sign the player in to **your own** account system.
2. Ask your server for `/api/platform/config`. If Baziness is off, skip everything below.
3. If no choice was made and the player isn't signed in, show the two buttons and wait.
4. Call `/api/platform/token`. If it answers `switchToDeviceId`, adopt that id and restart.

5. If the player just signed in, load their profile again, so the screen shows the signed-in account.

If the player closes the sign-in page without signing in, show the two buttons again.

Account switching, step by step

A switch happens when the Baziness account the player signs into is already linked to **another** of your players, for example after a reinstall or on a new phone:

1. Your server answers `switchToDeviceId: "<that player's device id>"`.
2. The client saves that device id as its own and forgets its stored Baziness token.
3. It restarts, without `linkedToken` in the address.
4. The next boot loads the other player, with all their progress.

Opening the game from the Baziness app

Important

The Baziness app now also sends `?linkCode=` (one-time). Forward it to your server instead of `linkedToken`. See [Token migration](#).

The Baziness app opens games with `?linkedToken=<token>` in the address:

- Send that token as `linkedToken` to `/api/platform/token`, so the player lands in the right Baziness account. Remove it from the address on the next restart.
- If the device already holds a **different** token, a different Baziness account is arriving: clear the local data and start again before continuing.
- Signing in decides **which** account plays, never **whether the player is new**. A first-time player arriving this way must still see your tutorial and receive your welcome gifts.

Signing in later, from Settings

A guest can connect their account at any time: Settings → open the **profile page** ([parameters](#)).

When the view closes, ask `/api/platform/token` again and refresh the coin balance.

Signing out

When the page sends `logout`, close the view, **delete everything your game stored locally** (identity, token, choices, tutorial progress) and restart as a brand-new visitor.

Testing logins on the dev platform

Tip

On the **development platform** (the one test games use), you can sign in with **any phone number** and the code `111111`. No SMS is sent. Make up numbers to create as many test accounts as you need: one for the account-switch test, one for a fresh guest, and so on. Production sends real SMS codes.

Common mistakes

Mistake	What happens	Fix
Treating the page's <code>token</code> message as a sign-in	the reward shop closes as soon as it opens; the first sign-in completes before the player signs in	only <code>login</code> is a sign-in
Linking a token without <code>validate-user-token</code>	anyone can claim any account	always verify on the server
Opening the sign-in page without the player's token	the first sign-in from the splash has no anonymous user to bind to	make sure <code>/api/platform/token</code> ran first and pass its token as <code>linkedToken</code>
No <code>allowGuest=false</code> on the sign-in page	the player sees a second "continue as guest"	add it on the sign-in, profile and offer pages
Rewarding the link more than once	signing out and in pays again	pay the "account connected" gift from the server, once per player

Ready-made code: [server endpoints](#), [client controller](#) and [messages](#).

Gita coins

How players earn and spend Gita coins, how your game shows the balance, gives coins safely, and treats players who haven't signed in yet.

How Gita coins work

Gita coin (گیتا کوین) is the Baziness currency. Players **earn** it in your game, for example by winning matches, ranking in tournaments, opening gifts or buying packs. They **spend** it on real prizes in the [reward shop](#). The balance lives **on the platform**, not in your database:

Action	Who	How
Show the balance	your client	<code>user/get-coin</code> with the player's user token
Give coins	your server	<code>client/add-coin</code> with the client token
Spend coins	the player	inside the reward shop page; your game does nothing

Warning

Never treat a locally stored number as the balance, and never let the client add coins.

Showing the balance

Refresh the balance:

- right after the client gets its token
- **when a Baziness page closes**, because the player may have spent coins in the reward shop
- when the game comes back to the foreground
- after your server gave coins

While the balance is unknown, or the call failed, show `...` or hide it. **Never show 0**: the player would think their coins are gone.

Giving coins safely

Every payout needs a **unique** `transactionId` **built from what the payout is**, for example `match:17` or `reward_claim:42` (the formula). Then:

- a retry, a double tap or a repeated request can never pay twice: the platform refuses the repeat with `400`
- treat that `400` as "already paid" and finish your own bookkeeping
- record every payout in **your own ledger** too (who, how much, why), for support and reports

Players who haven't signed in

A guest still has an anonymous Baziness user, so coins *can* be given to them. Many games still draw a line at **signed in**:

Kind of payout	Player not signed in	Why
Claims: a reward the player taps to collect (a win, a tournament prize, a gift)	refuse with a "sign in first" error; the client opens the sign-in page and retries after a successful sign-in	coins spent on real prizes should belong to a real account
Payouts that must not fail: a paid purchase that includes Gita coins	keep the coins in a local holding balance on your server	a paid order must never fail
When the player signs in	move the whole holding balance to Baziness with one <code>add-coin</code> (its own <code>transactionId</code>), then set the local balance to zero	the coins move exactly once

When Baziness is **off**, keep Gita coin rewards in that local balance as well, so your economy stays testable without credentials.

Showing the currency honestly

- Call it «گیتا کوین» everywhere, with its own icon.
- A build that ships **without** Baziness must not advertise Gita coin rewards.

Ready-made code: [giving coins](#), [showing the balance](#).

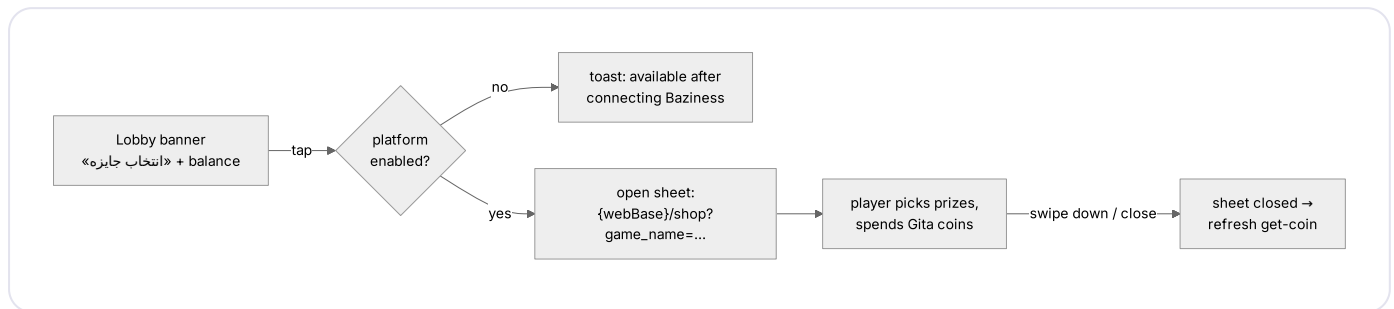
Reward page («انتخاب جایزه»)

Let players spend their Gita coins on real prizes — a button in your game that opens the Baziness reward shop.

What it is

«انتخاب جایزه» (*choose a prize*) is the Baziness **reward shop**: a page where players spend Gita coins on real prizes. Your game does three things:

1. **earns** coins for the player ([Gita coins](#))
2. **shows** the balance on a button or banner
3. **opens** the reward page in the in-game [web view](#) when the button is tapped



The address

Text

```
{web}/shop?game_name=<your game>&packageName=<package>&sdkType=js&linkedToken=<user token>&redirect=shop
```

The parameters are explained on [Baziness pages](#). Two rules matter most here:

- Open [/shop](#), not the site root. The root sends a player inside a game to the sign-in screen.
- **Don't** add [allowGuest=false](#): the shop is a destination, not a sign-in.

The button in your game

- Put it where players look: the lobby or the main menu. Show the **live balance** on it, with the Gita coin icon.
- When Baziness is off, either hide it or show "available after connecting to Baziness" when it is tapped.
- During your tutorial, the first tap can open your [game offer page](#) instead.

After the page closes

The player may have spent coins, so **refresh the balance** as soon as the view closes, and again when the game regains focus.

Hiding the platform's bottom bar (optional)

The Baziness site has its own bottom navigation (فروشگاه / بازی‌ها / خانه / ماموریت‌ها / بازی‌نس من). Inside your game you may want to hide it. The page belongs to another site, so your CSS and scripts can't reach into it. Instead:

1. Make the web view **taller** than its container by the bar's height.
2. Let the container **clip** what overflows.

The bar sits at the bottom of the page, so it falls outside the visible area while the rest of the page still scrolls. Tune the height on real phones. [Code](#).

Builds without Baziness

A build that ships without Baziness, for example some store builds, must not show the button **or** advertise Gita coin rewards. Let your server decide: it answers `{ enabled: false }` for that build, and the client never draws any of it.

Checklist

- ☐ The button shows the live balance (`...` while loading or on failure).
- ☐ It opens `/shop` with `game_name`, `packageName`, `sdkType=js`, `linkedToken` and `redirect=shop`.
- ☐ No `allowGuest=false`, and not the site root.
- ☐ The balance refreshes when the view closes and when the game regains focus.
- ☐ With Baziness off, the button is hidden or explains why it isn't available.

Tutorial offer page (game-offer)

A showcase page made for your game, opened once during the tutorial, that shows new players what Gita coins can buy and offers them a free starter pack.

What it is

Every game can have an **offer page** on Baziness, made for that game and selected by its `game_name` :

Text

```
{web}/game-offer?game_name=<your game>
```

It shows the prizes Gita coins can buy, and a special free pack for your game. For example, the example game's page shows «بسته ویژه بازی» example — «رایگان» with a «خرید رایگان» button. It is a **showcase**: you open it **once**, during the tutorial, to set expectations. After the tutorial, your reward button opens the real [reward shop](#).

Note

Ask the platform team to set up the offer page for your game, and use the same `game_name` as your reward page.

The address

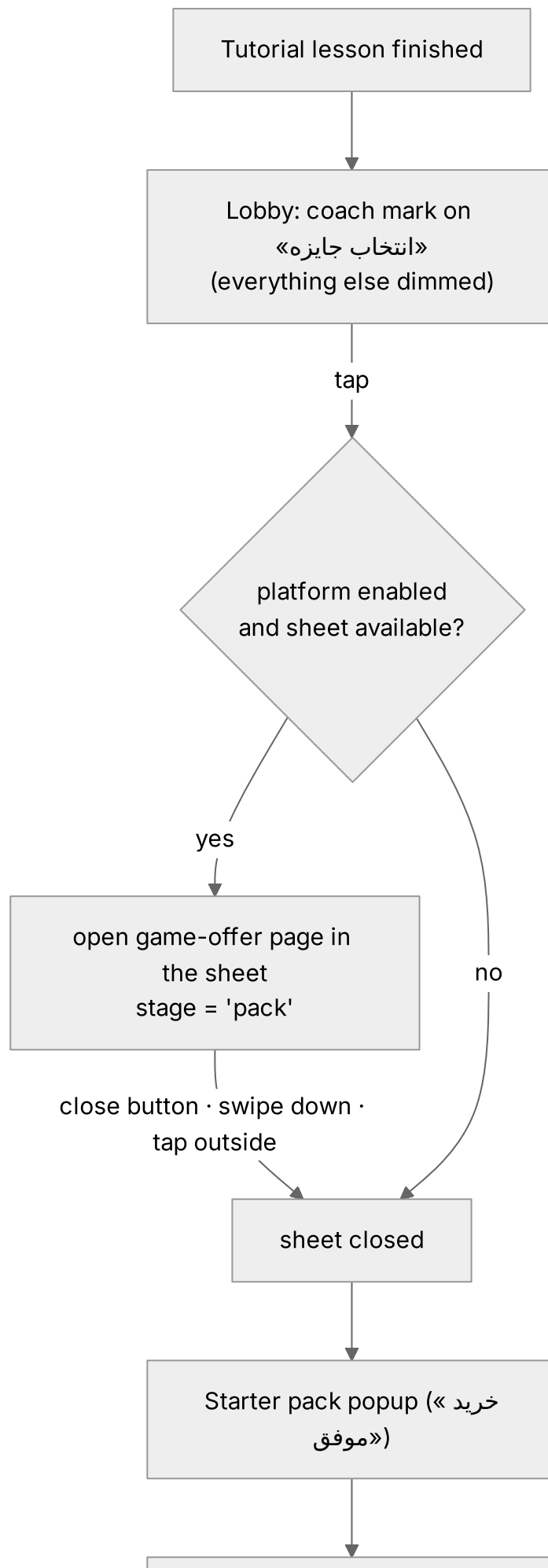
Environment	Address
Development	<code>https://platform-web-dev.karizmastudios.org/game-offer?game_name=<your game></code>
Production	<code>https://baziness.com/game-offer?game_name=<your game></code>

Like every Baziness page, it needs `packageName` and `sdkType=js` , plus `linkedToken` and `allowGuest=false` (parameters).

Warning

Without `sdkType=js` the offer page never sends its `close` message, so **its button can't close your view**.

The tutorial flow



Onboarding done →
the banner now opens
/shop

Rules

1. **Save the tutorial step before opening the page.** If the player reloads while the page is open, they continue with the starter pack and don't see the offer twice.
2. **Continue however the page was closed:** its own button, a swipe down, or a tap outside.
3. **Never block the tutorial on Baziness.** If it is off or unreachable, skip straight to the starter pack.
4. **Give the starter pack from the server, once per player.** Present it as a successful purchase («خريد موفق»), even though it is free.
5. **Players arriving from the Baziness app get the tutorial too.** Signing in doesn't make a player "old".
6. **After the tutorial, open /shop**, never the offer page again.

Test it

1. Clear the site data (or use a private window) and open your game.
2. Choose either option on the splash, and finish the tutorial's first lesson.
3. Only the reward button is highlighted. Tap it: the offer page slides up.
4. Close it each way in turn (its button, a swipe on the handle, a tap outside). Every way must bring the starter pack.
5. Tap the reward button again: now the **real /shop** opens.

Ready-made code: [client](#). See it live in the [example game](#).

Links & UTM parameters

The three kinds of link in a Baziness integration, and how to tag the links that bring players to your game so every player is credited to the right campaign.

Three kinds of link

Link	Direction	Built by	Details
Baziness page links (sign-in, profile, <code>/shop</code> , <code>/game-offer</code>)	game → Baziness	your client, with settings from your server	Baziness pages
Payment page link (<code>/buy</code>)	game → Baziness Pay	your server only	Baziness Pay
Links into your game (<code>?utm_...</code> , <code>?tag=</code> , <code>?linkedToken=</code>)	outside → game	marketing, ads, the Baziness app	this page

Links into your game

Every link that brings players to your game should say **where they came from**:

Text

```
https://yourgame.example.com/?
utm_source=baziness&utm_medium=inapp&utm_campaign=shahrivar_1405&utm_content=banner_home
https://yourgame.example.com/?tag=telegram
```

Parameter	Meaning	Examples
<code>utm_source</code>	who sends the traffic	<code>baziness</code> , <code>telegram</code> , <code>instagram</code>
<code>utm_medium</code>	the kind of channel	<code>inapp</code> , <code>banner</code> , <code>cpc</code> , <code>social</code> , <code>push</code>
<code>utm_campaign</code>	the campaign	<code>shahrivar_1405</code> , <code>nowruz_1405</code>
<code>utm_content</code>	the creative or placement	<code>banner_home</code> , <code>story_2</code>
<code>tag</code>	a short channel tag (simpler alternative)	<code>telegram</code> , <code>google</code>
<code>linkedToken</code>	added by the Baziness app when it opens your game	the player's user token (Login)

Important

UTM values must be **plain ASCII**: lowercase letters, digits, `.` , `_` , `-` , at most 64 characters. A Persian campaign name is lost, so write `shahrivar_1405` , not `شهریور` .

How your game should read them

1. **Read them first**, before anything renders. Match the names case-insensitively.
2. **Remove them from the address bar**. The player doesn't see or edit them, and a player sharing their address doesn't credit your campaign with their friends.
3. **Store them on the device the first time** ("first touch"), and only if nothing is stored yet. A Baziness sign-in or an account switch restarts the game, and without local storage the tags would be lost.
4. **Send them with the first request that creates the player**, both your normal sign-in and `/api/platform/token` , whichever runs first.
5. **On the server, save them only when the player is created**. Never overwrite them later.
6. **Keep four separate fields** (`utm_source` , `utm_medium` , `utm_campaign` , `utm_content`), not one joined string. Campaign names contain underscores, so a joined value can't be split back.
7. **Leave `linkedToken` alone**. It isn't a campaign tag: the sign-in logic uses it, and removes it on the next restart.

Making campaign links

Give marketing a fixed recipe, or a small link generator, so every link carries **all four** parameters spelled the same way: your game's address, plus `utm_source`, `utm_medium`, `utm_campaign` and (optionally) `utm_content`, all in lowercase ASCII.

Test your links

- Open a campaign link in a private window. The address bar must be clean after the page loads.
- Sign in with Baziness, or trigger an account switch. The new player must still carry the campaign.
- Open the same link with an **existing** player. Their stored campaign must not change.

Ready-made code: [reading the tags and saving them](#), [building links](#).

Migrating to link codes & short-lived tokens

The platform is replacing long-lived user tokens with 15-minute tokens and one-time link codes. What changes for your server and your client, step by step, and by when.

What is changing and why

Until now, a Baziness **user token** was valid for **10 years**. It also travelled in addresses:

- your game opened Baziness pages with `?linkedToken=<token>` ;
- the Baziness app opened your game with `?linkedToken=<token>` .

Addresses end up in server logs, analytics, browser history and Referer headers. A token found there gave access to the player's coins and account for a decade, and signing out couldn't stop it.

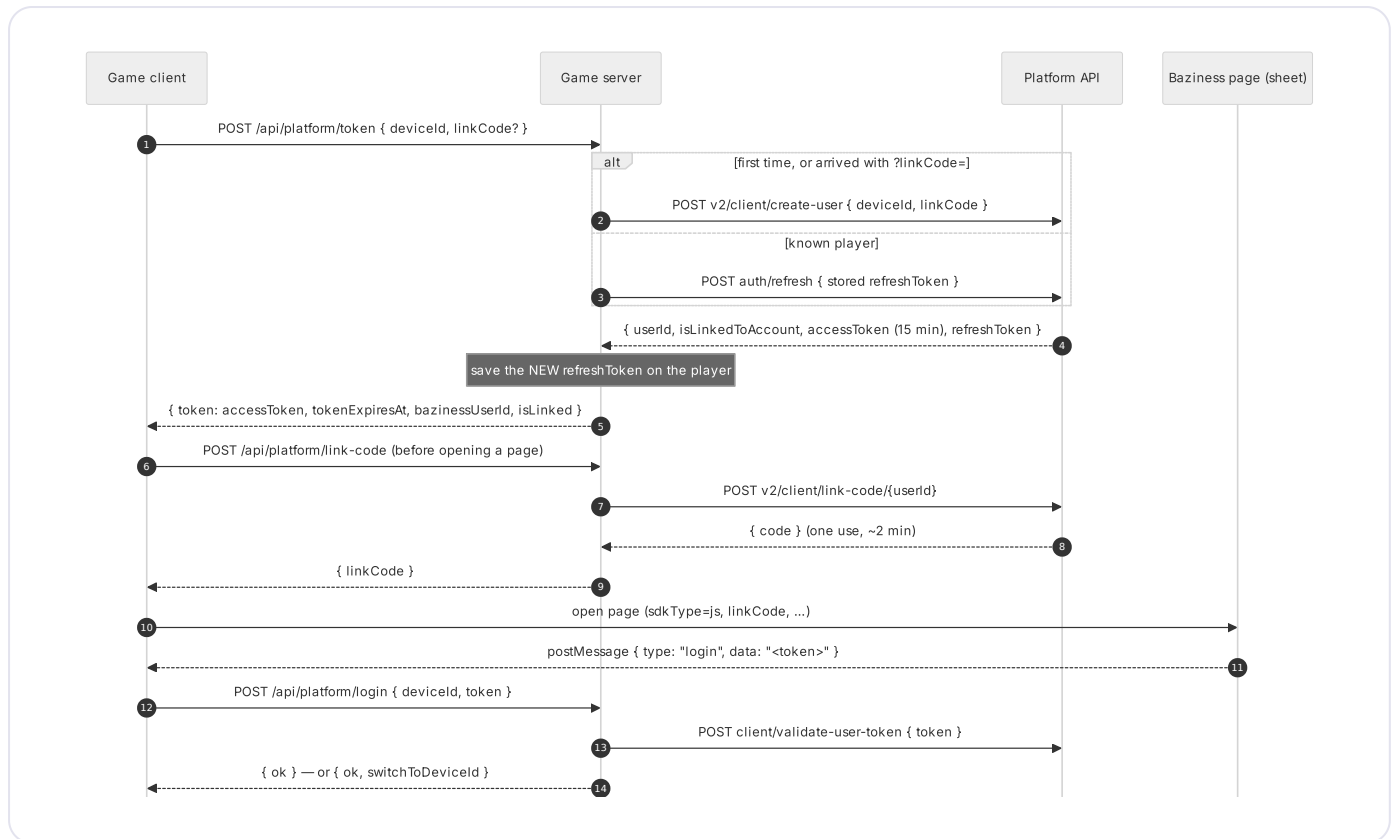
The new model:

	Before	Now
Access token (what your client sends as <code>Bearer</code>)	valid 10 years	valid 15 minutes
Refresh token	—	renews the access token; kept on your server , never on the device; single use, so every renewal returns a new one
Moving a player between apps	<code>?linkedToken=<token></code>	<code>?linkCode=<code></code> : a one-time code , valid about 2 minutes , useless after first use

Important

Both models work side by side during the migration. The old endpoints and `?linkedToken=` keep working, and the Baziness app sends both `?linkCode=` and `?linkedToken=` to games. The platform team will announce a **cut-off date**: after it, old tokens and `linkedToken` stop working. Plan the migration before then.

The flow after the migration:



What changes at a glance

Today	After the migration	Where
<code>client/create-user</code> with <code>linkedToken</code>	<code>v2/client/create-user</code> with <code>linkCode</code> ; save <code>tokens.refreshToken</code>	server
<code>client/get-user-token/{id}</code> on every boot	<code>auth/refresh</code> with the saved refresh token; <code>v2/client/get-user-token/{id}</code> only when there is no session	server
open pages with <code>linkedToken=<token></code>	open pages with <code>linkCode=<code></code> , a new code from <code>v2/client/link-code/{id}</code> for each page	server + client
the Baziness app opens your game with <code>?linkedToken=</code>	<code>?linkCode=</code> : forward it to your server like the token before	client
<code>GET client/validate-user-token/{token}</code>	<code>POST client/validate-user-token { token }</code> (the token is no longer in the address)	server
the client keeps its token for years	the client keeps the access token only; when it expires, it asks your server again	client

Nothing else changes: the page messages (`login` , `token` , `close` , `logout`), coins, rewards and payments work as before.

Server, step by step

1. Keep a refresh token per player

Add a column to your player record, for example `business_refresh_token` . Treat it like a password:

- never log it;
- never send it to the client;
- encrypt it if you encrypt other secrets.

Always overwrite it with the newest one. Every `create-user` , `get-user-token` and `refresh` returns a new refresh token, and the previous one stops working.

2. `/api/platform/token` : refresh instead of `get-user-token`

This is the endpoint your client calls on boot. The client now sends `linkCode` instead of `linkedToken` .

1. **The client sent a `linkCode` , or the player has no Baziness user yet:** call `v2/client/create-user` with `deviceId` and `linkCode` . Then handle the returned `userId` as you do today:
 - The `userId` belongs to **another** of your players → answer `switchToDeviceId` (same person, another device).
 - The player **already has a different** `userId` → a different Baziness account arrived. Start it as its own player and answer `switchToDeviceId` (the old "clear local data and start again" rule).
 - Otherwise → save `userId` and the refresh token on this player.
 - If the code was **expired or already used** (`400 invalid_link_code`), continue as if no code was sent.
2. **A known player:** call `auth/refresh` with the saved refresh token, and save the new one. The answer also contains `isLinkedToAccount` , so you still learn when the player signed in.
 - `401 refresh_token_rotated` : another request of the same player refreshed a moment earlier. **Reload** the refresh token from your database and try once more.
 - Any other `401` (`session_revoked` , `refresh_token_expired` , `invalid_refresh_token` , `refresh_token_reused`): the session is over. Call `v2/client/get-user-token/{userId}` for a new one and save its refresh token.
 - `403 account_suspended` : the account is blocked.
3. **Answer the client** with the **access token** and its expiry: `{ token, tokenExpiresAt, businessUserId, isLinkedToAccount }` . Never include the refresh token.

Tip

Players you already have don't hold a refresh token yet. Step 2 handles them on its own: no saved refresh token → `v2/client/get-user-token` once, and from then on `auth/refresh`.

3. New `/api/platform/link-code` : a code for each page

When the client is about to open a Baziness page, it asks your server for a code. Your server calls `v2/client/link-code/{userId}` and returns `{ linkCode }`.

- A code works **once** and for about **two minutes**: get a new one for every page you open.
- It only works for Baziness pages. Even if it leaks, nobody can use it to log in to your game.
- Protect the endpoint like your other player endpoints: only the player themselves may ask for their code.

4. `/api/platform/login` : verify with POST

Nothing changes in the logic. Only the call does: `POST client/validate-user-token` with `{ token }` in the body.

The token in the `login` message is now a **15-minute** token. Verify it right away, as you already do.

If the token belongs to a different Baziness user than the player's current one, clear the player's saved refresh token; the next `/token` call starts a new session.

5. New `/api/platform/logout`

When a page sends `logout`, the client calls this before starting over. Your server ends the session (`auth/logout`) and clears the saved refresh token.

Server code

Additions to your platform client (next to `createUser` / `getUserAccessToken`):

TypeScript (Node)

```
// platform/baziness.ts - ADD these next to createUser / getUserAccessToken (they reuse
API_BASE and platformFetch).

/** A player's session: the access token goes to your client, the refresh token stays in your
database. */
export interface PlatformSession {
  userId: number;
  isLinkedToAccount: boolean;
  accessToken: string;
  accessTokenExpiresAt: string;
  refreshToken: string;
}
```

```

const readSession = (j: Record<string, any>): PlatformSession => {
  const t = j.tokens ?? j; // v2 create-user / get-user-token nest the pair in `tokens`;
  auth/refresh returns it flat
  return {
    userId: Number(j.userId),
    isLinkedToAccount: Boolean(j.isLinkedToAccount),
    accessToken: String(t.accessToken),
    accessTokenExpiresAt: String(t.accessTokenExpiresAt),
    refreshToken: String(t.refreshToken),
  };
};

/** First time, or when the client brought a ?linkCode=. Throws (status 400) for an expired
or used code. */
export const createUserSession = async (deviceId: string, platform: string, store: string,
linkCode: string | null) =>
  readSession(await platformFetch("POST", "/v2/client/create-user", { deviceId, platform,
store, linkCode }));

/** A new session for a player you know – only when you have no refresh token or the session
ended. */
export const startUserSession = async (baziinessUserId: number) =>
  readSession(await platformFetch("POST", `/v2/client/get-user-token/${baziinessUserId}`));

/**
 * Renews a session. Returns the new session, "rotated" when a parallel request refreshed
first
 * (reload the stored token and retry once), or null when the session is over (start a new
one).
 */
export async function refreshUserSession(refreshToken: string): Promise<PlatformSession |
"rotated" | null> {
  const res = await fetch(`${API_BASE}/auth/refresh`, { // no client token: the refresh
token is the credential
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ refreshToken }),
  });
  if (res.ok) return readSession(await res.json());
  const error = ((await res.json()).catch(() => ({}))) as { error?: string }.error;
  if (res.status === 401 && error === "refresh_token_rotated") return "rotated";
  if (res.status === 403) throw Object.assign(new Error("account_suspended"), { status: 403
});
  if (res.status === 401) return null;
  throw new Error(`auth/refresh failed (${res.status})`); // 5xx / network: keep the old
session, try later
}

/** Ends a session (player signed out). Never throws. */
export async function endUserSession(refreshToken: string): Promise<void> {
  await fetch(`${API_BASE}/auth/logout`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ refreshToken }),
  }).catch(() => {});
}

/** One-time code for opening ONE Baziiness page as this player (~2 minutes). */
export const createPageLinkCode = async (baziinessUserId: number) =>

```

```
String((await platformFetch("POST", `/v2/client/link-code/${baziinessUserId}`).code));

/** REPLACES the old GET version: the token now goes in the body, not in the URL. */
export async function validateUserAccessToken(token: string): Promise<number> {
  const j = await platformFetch("POST", "/client/validate-user-token", { token });
  if (!(j.isValid ?? j.IsValid)) throw new Error("invalid user token");
  return Number(j.userId ?? j.UserId ?? 0);
}
```

C# (.NET)

```
// BaziinessPlatformClient.cs – ADD these members (they reuse settings, http, SendAsync and
Json).

public record PlatformSession(long UserId, bool IsLinkedToAccount, string AccessToken,
    DateTimeOffset AccessTokenExpiresAt, string RefreshToken);

public enum RefreshOutcome { Ok, Rotated, Ended }

static PlatformSession ReadSession(JsonElement j)
{
    // v2 create-user / get-user-token nest the pair in "tokens"; auth/refresh returns it
    flat.
    var t = j.TryGetProperty("tokens", out var nested) ? nested : j;
    return new PlatformSession(
        j.GetProperty("userId").GetInt64(),
        j.GetProperty("isLinkedToAccount").GetBoolean(),
        t.GetProperty("accessToken").GetString()!,
        t.GetProperty("accessTokenExpiresAt").GetDateTimeOffset(),
        t.GetProperty("refreshToken").GetString()!);
}

/// <summary>First time, or when the client brought a ?linkCode=. 400 = expired or used code.
</summary>
public async Task<PlatformSession> CreateUserSessionAsync(string deviceId, string platform,
    string store, string? linkCode) =>
    ReadSession(await SendAsync(HttpMethod.Post, "/v2/client/create-user", new { deviceId,
    platform, store, linkCode }));

/// <summary>A new session for a known player – only when there is no refresh token or the
session ended.</summary>
public async Task<PlatformSession> StartUserSessionAsync(long baziinessUserId) =>
    ReadSession(await SendAsync(HttpMethod.Post, $"/v2/client/get-user-
token/{baziinessUserId}"));

/// <summary>Renews a session. Rotated = a parallel request refreshed first (reload, retry
once); Ended = start a new one.</summary>
public async Task<(RefreshOutcome Outcome, PlatformSession? Session)>
RefreshUserSessionAsync(string refreshToken)
{
    // No client token here: the refresh token itself is the credential.
    var res = await http.PostAsJsonAsync($"{settings.ApiBase}/auth/refresh", new {
refreshToken }, Json);
    if (res.IsSuccessStatusCode)
        return (RefreshOutcome.Ok, ReadSession(await
res.Content.ReadFromJsonAsync<JsonElement>(Json)));

    string? error = null;
    var text = await res.Content.ReadAsStringAsync();
```

```

    if (!string.IsNullOrEmpty(text))
    {
        try { error = JsonDocument.Parse(text).RootElement.TryGetProperty("error", out var e)
? e.GetString() : null; }
        catch (JsonException) { /* not JSON */ }
    }
    return res.StatusCode switch
    {
        HttpStatusCode.Unauthorized when error == "refresh_token_rotated" =>
(RefreshOutcome.Rotated, null),
        HttpStatusCode.Unauthorized => (RefreshOutcome.Ended, null),
        HttpStatusCode.Forbidden => throw new InvalidOperationException("account_suspended"),
        _ => throw new HttpRequestException($"auth/refresh failed ({(int)res.StatusCode}")
// keep the old session
    };
}

/// <summary>Ends a session (player signed out). Never throws.</summary>
public async Task EndUserSessionAsync(string refreshToken)
{
    try { await http.PostAsJsonAsync($"{settings.ApiBase}/auth/logout", new { refreshToken },
Json); }
    catch { /* best effort */ }
}

/// <summary>One-time code for opening ONE Baziness page as this player (~2 minutes).
</summary>
public async Task<string> CreatePageLinkCodeAsync(long bazinessUserId) =>
    (await SendAsync(HttpMethod.Post, $"/v2/client/link-
code/{bazinessUserId}")).GetProperty("code").GetString()!;

/// <summary>REPLACES the old GET version: the token now goes in the body, not in the URL.
</summary>
public async Task<long> ValidateUserAccessTokenAsync(string token)
{
    var j = await SendAsync(HttpMethod.Post, "/client/validate-user-token", new { token });
    if (!j.GetProperty("isValid").GetBoolean()) throw new InvalidOperationException("invalid
user token");
    return j.GetProperty("userId").GetInt64();
}

```

Python

```

# baziness/platform.py - ADD these methods to BazinessPlatform (they reuse API_BASE,
self._http and self._call).

@staticmethod
def _session(j: dict) -> dict:
    t = j.get("tokens") or j # v2 create-user / get-user-token nest the pair;
auth/refresh returns it flat
    return {"userId": int(j["userId"]), "isLinkedToAccount":
bool(j["isLinkedToAccount"]),
            "accessToken": t["accessToken"], "accessTokenExpiresAt":
t["accessTokenExpiresAt"],
            "refreshToken": t["refreshToken"]}

def create_user_session(self, device_id: str, platform: str, store: str, link_code: str |
None) -> dict:
    """First time, or when the client brought a ?linkCode=. HTTPError 400 = expired or

```

```

used code."""
    return self._session(self._call("POST", "/v2/client/create-user", {
        "deviceId": device_id, "platform": platform, "store": store, "linkCode":
link_code}))

    def start_user_session(self, baziness_user_id: int) -> dict:
        """A new session for a known player – only when there is no refresh token or the
session ended."""
        return self._session(self._call("POST", f"/v2/client/get-user-
token/{baziness_user_id}"))

    def refresh_user_session(self, refresh_token: str):
        """Returns the new session, "rotated" (reload the stored token, retry once) or None
(session over)."""
        r = self._http.post(f"{API_BASE}/auth/refresh", json={"refreshToken": refresh_token},
timeout=10)
        if r.ok:
            return self._session(r.json())
        error = (r.json() if r.content else {}).get("error")
        if r.status_code == 401:
            return "rotated" if error == "refresh_token_rotated" else None
        if r.status_code == 403:
            raise PermissionError("account_suspended")
        r.raise_for_status() # 5xx: keep the old session, try later

    def end_user_session(self, refresh_token: str) -> None:
        """Ends a session (player signed out). Never raises."""
        try:
            self._http.post(f"{API_BASE}/auth/logout", json={"refreshToken": refresh_token},
timeout=10)
        except requests.RequestException:
            pass

    def create_page_link_code(self, baziness_user_id: int) -> str:
        """One-time code for opening ONE Baziness page as this player (~2 minutes)."""
        return str(self._call("POST", f"/v2/client/link-code/{baziness_user_id}")[ "code"])

    def validate_user_access_token(self, token: str) -> int:
        """REPLACES the old GET version: the token now goes in the body, not in the URL."""
        j = self._call("POST", "/client/validate-user-token", {"token": token})
        if not _pick(j, "isValid", False):
            raise ValueError("invalid user token")
        return int(_pick(j, "userId", 0))

```

The four endpoints:

TypeScript (Node)

```

import {
    createUserSession, startUserSession, refreshUserSession, endUserSession,
createPageLinkCode,
    validateUserAccessToken, platformClientConfig, platformEnabled, type PlatformSession,
} from "../platform/baziness";
import { db } from "../db"; // your own data layer. New column on the player:
baziness_refresh_token (a secret)

// What the client gets. The refresh token never leaves your server.
const forClient = (s: PlatformSession) => ({

```

```

    token: s.accessToken, tokenExpiresAt: s.accessTokenExpiresAt,
    bazinessUserId: s.userId, isLinkedToAccount: s.isLinkedToAccount,
  });

  async function keep(playerId: number, s: PlatformSession) {
    await db.setRefreshToken(playerId, s.refreshToken); // always the NEWEST one: the old one stops working
    return s;
  }

  /** A valid session for a player who already has a Baziness user. */
  async function sessionFor(player: { id: number; bazinessUserId: number; bazinessRefreshToken: string | null }) {
    if (player.bazinessRefreshToken) {
      let r = await refreshUserSession(player.bazinessRefreshToken);
      if (r === "rotated") { // another request of this player refreshed a moment ago
        const latest = await db.getRefreshToken(player.id);
        r = latest ? await refreshUserSession(latest) : null;
        if (r === "rotated") r = null;
      }
      if (r) return keep(player.id, r);
    }
    // No refresh token yet (players from before the migration) or the session ended.
    return keep(player.id, await startUserSession(player.bazinessUserId));
  }

  app.post("/api/platform/token", async (req, res) => {
    const deviceId = String(req.body.deviceId ?? "").trim();
    const linkCode: string | null = req.body.linkCode ?? null; // replaces linkedToken
    if (!deviceId) return res.status(400).json({ error: "deviceId required" });
    if (!platformEnabled()) return res.json({ enabled: false });

    const config = { enabled: true, ...platformClientConfig() };
    const me = await db.ensureUserByDevice(deviceId, { utm: req.body.utm, tag: req.body.tag });
    try {
      if (linkCode || !me.bazinessUserId) {
        // First time, or arrived from the Baziness app with ?linkCode=: which Baziness user is this?
        const s = await createUserSession(deviceId, "Web", "Direct", linkCode)
          .catch((e) => (linkCode && e.status === 400 ? null : Promise.reject(e))); // expired or used code
        if (s) {
          const owner = await db.findUserByBazinessId(s.userId);
          if (owner && owner.id !== me.id) { // the same person, another player of yours
            await keep(owner.id, s);
            return res.json({ ...config, switchToDeviceId: owner.deviceId });
          }
          if (me.bazinessUserId && me.bazinessUserId !== s.userId) {
            // A different Baziness account arrived on a device that already has a player: // it becomes its own player (the old "clear local data and start again" rule).
            const fresh = await db.createPlayerForBaziness(s.userId, s.isLinkedToAccount);
            await keep(fresh.id, s);
            return res.json({ ...config, switchToDeviceId: fresh.deviceId });
          }
        }
        await db.linkBaziness(me.id, s.userId, s.isLinkedToAccount);
        await keep(me.id, s);
        return res.json({ ...config, ...forClient(s) });
      }
    }
  });

```

```

    }
    if (!me.bazinessUserId) { // stale code and no Baziness
user yet
        const s2 = await createUserSession(deviceId, "Web", "Direct", null);
        await db.linkBaziness(me.id, s2.userId, s2.isLinkedToAccount);
        await keep(me.id, s2);
        return res.json({ ...config, ...forClient(s2) });
    }
}
const s = await sessionFor(me);
if (s.isLinkedToAccount !== me.isBazinessLinked) await db.setLinked(me.id,
s.isLinkedToAccount);
return res.json({ ...config, ...forClient(s) });
} catch (err) {
    console.error("[platform] token failed:", (err as Error).message);
    const suspended = (err as { status?: number }).status === 403;
    return res.json({ ...config, error: suspended ? "account_suspended" :
"platform_unavailable" }); // never block the boot
}
});

// Protect it like your other player endpoints: only the player themself may ask for their
code.
app.post("/api/platform/link-code", async (req, res) => {
    const me = await db.findUserByDevice(String(req.body.deviceId ?? ""));
    if (!me?.bazinessUserId) return res.status(409).json({ error: "call /api/platform/token
first" });
    try {
        return res.json({ linkCode: await createPageLinkCode(me.bazinessUserId) });
    } catch {
        return res.status(503).json({ error: "platform_unavailable" });
    }
});

app.post("/api/platform/login", async (req, res) => {
    const deviceId = String(req.body.deviceId ?? "").trim();
    const token = String(req.body.token ?? "").trim();
    if (!deviceId || !token) return res.status(400).json({ error: "deviceId and token required"
});
    if (!platformEnabled()) return res.json({ ok: false, reason: "disabled" });
    try {
        const bazinessUserId = await validateUserAccessToken(token); // POST now; verify
right away (15-min token)
        const owner = await db.findUserByBazinessId(bazinessUserId);
        const me = await db.ensureUserByDevice(deviceId);
        if (owner && owner.id !== me.id) return res.json({ ok: true, switchToDeviceId:
owner.deviceId });
        if (me.bazinessUserId !== bazinessUserId) await db.setRefreshToken(me.id, null); //
belonged to the old user
        await db.linkBaziness(me.id, bazinessUserId, true);
        return res.json({ ok: true }); // the client then
calls /token again
    } catch {
        return res.json({ ok: false, reason: "invalid" });
    }
});

// The page sent `logout`: end the session before the client starts over as a new visitor.
app.post("/api/platform/logout", async (req, res) => {
    const me = await db.findUserByDevice(String(req.body.deviceId ?? ""));

```

```

    if (me?.baziinessRefreshToken) {
        await endUserSession(me.baziinessRefreshToken);
        await db.setRefreshToken(me.id, null);
    }
    return res.json({ ok: true });
});

```

C# (.NET)

```

// IUserStore gains: SetRefreshTokenAsync(id, token?), GetRefreshTokenAsync(id),
// FindByDeviceAsync(deviceId),
// CreatePlayerForBaziinessAsync(baziinessUserId, isLinked). Store the
// refresh token like a password.

// What the client gets. The refresh token never leaves your server.
static object ForClient(PlatformSettings pub, PlatformSession s) => new
{
    enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
    token = s.AccessToken, tokenExpiresAt = s.AccessTokenExpiresAt,
    baziinessUserId = s.UserId, isLinkedToAccount = s.IsLinkedToAccount
};

static async Task<PlatformSession> SessionForAsync(Player player, BaziinessPlatformClient
platform, IUserStore users)
{
    var refreshToken = player.BaziinessRefreshToken;
    for (var attempt = 0; refreshToken is not null && attempt < 2; attempt++)
    {
        var (outcome, session) = await platform.RefreshUserSessionAsync(refreshToken);
        if (outcome == RefreshOutcome.Ok)
        {
            await users.SetRefreshTokenAsync(player.Id, session!.RefreshToken); // always the
newest one
            return session;
        }
        if (outcome == RefreshOutcome.Ended) break;
        refreshToken = await users.GetRefreshTokenAsync(player.Id); // Rotated: a
parallel request won
    }
    // No refresh token yet (players from before the migration) or the session ended.
    var fresh = await platform.StartUserSessionAsync(player.BaziinessUserId!.Value);
    await users.SetRefreshTokenAsync(player.Id, fresh.RefreshToken);
    return fresh;
}

app.MapPost("/api/platform/token", async (TokenRequest body, BaziinessPlatformClient platform,
IUserStore users, PlatformSettings pub, ILogger<Program> log) =>
{
    if (string.IsNullOrWhiteSpace(body.DeviceId)) return Results.BadRequest(new { error =
"deviceId required" });
    if (!pub.IsOn) return Results.Ok(new { enabled = false });

    var me = await users.EnsureByDeviceAsync(body.DeviceId);
    try
    {
        if (body.LinkCode is not null || me.BaziinessUserId is null)
        {
            PlatformSession? s = null;
            try { s = await platform.CreateUserSessionAsync(body.DeviceId, "Web", "Direct",

```

```

body.LinkCode); }
    catch (HttpRequestException e) when (body.LinkCode is not null && e.StatusCode ==
HttpStatusCode.BadRequest) { }

    if (s is not null)
    {
        var owner = await users.FindByBazinessUserIdAsync(s.UserId);
        if (owner is not null && owner.Id != me.Id)
        {
            await users.SetRefreshTokenAsync(owner.Id, s.RefreshToken);
            return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase,
pub.PackageName, switchToDeviceId = owner.DeviceId });
        }
        if (me.BazinessUserId is not null && me.BazinessUserId != s.UserId)
        {
            var created = await users.CreatePlayerForBazinessAsync(s.UserId,
s.IsLinkedToAccount);
            await users.SetRefreshTokenAsync(created.Id, s.RefreshToken);
            return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase,
pub.PackageName, switchToDeviceId = created.DeviceId });
        }
        await users.LinkBazinessAsync(me.Id, s.UserId, s.IsLinkedToAccount);
        await users.SetRefreshTokenAsync(me.Id, s.RefreshToken);
        return Results.Ok(ForClient(pub, s));
    }
    if (me.BazinessUserId is null) // stale code and no Baziness
user yet
    {
        var s2 = await platform.CreateUserSessionAsync(body.DeviceId, "Web",
"Direct", null);
        await users.LinkBazinessAsync(me.Id, s2.UserId, s2.IsLinkedToAccount);
        await users.SetRefreshTokenAsync(me.Id, s2.RefreshToken);
        return Results.Ok(ForClient(pub, s2));
    }

    var session = await SessionForAsync(me, platform, users);
    if (session.IsLinkedToAccount != me.IsBazinessLinked) await
users.SetLinkedAsync(me.Id, session.IsLinkedToAccount);
    return Results.Ok(ForClient(pub, session));
}
catch (Exception ex)
{
    log.LogError(ex, "platform token failed");
    var error = ex.Message == "account_suspended" ? "account_suspended" :
"platform_unavailable";
    return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
error }); // never block the boot
}
});

// Protect it like your other player endpoints: only the player themself may ask for their
code.
app.MapPost("/api/platform/link-code", async (DeviceRequest body, BazinessPlatformClient
platform, IUserStore users) =>
{
    var me = await users.FindByDeviceAsync(body.DeviceId);
    if (me?.BazinessUserId is null) return Results.Conflict(new { error = "call
/api/platform/token first" });
    try { return Results.Ok(new { linkCode = await

```

```

platform.CreatePageLinkCodeAsync(me.BazinessUserId.Value) }); }
    catch { return Results.StatusCode(503); }
});

app.MapPost("/api/platform/login", async (LoginRequest body, BazinessPlatformClient platform,
IUserStore users, PlatformSettings pub) =>
{
    if (string.IsNullOrEmpty(body.DeviceId) || string.IsNullOrEmpty(body.Token))
        return Results.BadRequest(new { error = "deviceId and token required" });
    if (!pub.IsOn) return Results.Ok(new { ok = false, reason = "disabled" });
    try
    {
        long bazinessUserId = await platform.ValidateUserAccessTokenAsync(body.Token); //
        POST now; verify right away
        var owner = await users.FindByBazinessUserIdAsync(bazinessUserId);
        var me = await users.EnsureByDeviceAsync(body.DeviceId);
        if (owner is not null && owner.Id != me.Id) return Results.Ok(new { ok = true,
switchToDeviceId = owner.DeviceId });
        if (me.BazinessUserId != bazinessUserId) await users.SetRefreshTokenAsync(me.Id,
null); // belonged to the old user
        await users.LinkBazinessAsync(me.Id, bazinessUserId, isLinked: true);
        return Results.Ok(new { ok = true }); //
        the client then calls /token again
    }
    catch { return Results.Ok(new { ok = false, reason = "invalid" }); }
});

// The page sent `logout`: end the session before the client starts over as a new visitor.
app.MapPost("/api/platform/logout", async (DeviceRequest body, BazinessPlatformClient
platform, IUserStore users) =>
{
    var me = await users.FindByDeviceAsync(body.DeviceId);
    if (me?.BazinessRefreshToken is { } refreshToken)
    {
        await platform.EndUserSessionAsync(refreshToken);
        await users.SetRefreshTokenAsync(me.Id, null);
    }
    return Results.Ok(new { ok = true });
});

public record TokenRequest(string DeviceId, string? LinkCode);
public record DeviceRequest(string DeviceId);
public record LoginRequest(string DeviceId, string Token);

```

Python

```

from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from requests import HTTPError
from baziness.platform import platform, platform_client_config, platform_enabled
from app import db # your own data layer. New column on the player: baziness_refresh_token
(a secret)

app = FastAPI()

class TokenRequest(BaseModel):
    deviceId: str
    linkCode: str | None = None # replaces linkedToken

```

```

class DeviceRequest(BaseModel):
    deviceId: str

class LoginRequest(BaseModel):
    deviceId: str
    token: str

def for_client(s: dict) -> dict: # the refresh token never leaves your server
    return {"token": s["accessToken"], "tokenExpiresAt": s["accessTokenExpiresAt"],
            "baziinessUserId": s["userId"], "isLinkedToAccount": s["isLinkedToAccount"]}

def keep(player_id: int, s: dict) -> dict:
    db.set_refresh_token(player_id, s["refreshToken"]) # always the newest one
    return s

def session_for(player) -> dict:
    token = player.baziiness_refresh_token
    for _ in range(2):
        if not token:
            break
        r = platform.refresh_user_session(token)
        if isinstance(r, dict):
            return keep(player.id, r)
        if r is None: # the session ended
            break
        token = db.get_refresh_token(player.id) # "rotated": a parallel request won
    # No refresh token yet (players from before the migration) or the session ended.
    return keep(player.id, platform.start_user_session(player.baziiness_user_id))

@app.post("/api/platform/token")
def platform_token(body: TokenRequest):
    if not platform_enabled():
        return {"enabled": False}
    config = {"enabled": True, **platform_client_config()}
    me = db.ensure_user_by_device(body.deviceId)
    try:
        if body.linkCode or not me.baziiness_user_id:
            try:
                s = platform.create_user_session(body.deviceId, "Web", "Direct",
body.linkCode)
            except HTTPError as e:
                if not (body.linkCode and e.response.status_code == 400):
                    raise
                s = None # expired or used code
            if s:
                owner = db.find_user_by_baziiness_user_id(s["userId"])
                if owner and owner.id != me.id: # the same person, another player of
yours
                    keep(owner.id, s)
                    return {**config, "switchToDeviceId": owner.device_id}
                if me.baziiness_user_id and me.baziiness_user_id != s["userId"]:
                    fresh = db.create_player_for_baziiness(s["userId"],
s["isLinkedToAccount"])

```

```

        keep(fresh.id, s)
        return {**config, "switchToDeviceId": fresh.device_id}
    db.link_baziness(me.id, s["userId"], s["isLinkedToAccount"])
    return {**config, **for_client(keep(me.id, s))}
    if not me.baziness_user_id:
        # stale code and no Baziness user yet
        s2 = platform.create_user_session(body.deviceId, "Web", "Direct", None)
        db.link_baziness(me.id, s2["userId"], s2["isLinkedToAccount"])
        return {**config, **for_client(keep(me.id, s2))}
    s = session_for(me)
    if s["isLinkedToAccount"] != me.is_baziness_linked:
        db.set_linked(me.id, s["isLinkedToAccount"])
    return {**config, **for_client(s)}
except PermissionError:
    return {**config, "error": "account_suspended"}
except Exception as err: # never block the boot
    print("[platform] token failed:", err)
    return {**config, "error": "platform_unavailable"}

@app.post("/api/platform/link-code") # protect it like your other player endpoints
def platform_link_code(body: DeviceRequest):
    me = db.find_user_by_device(body.deviceId)
    if not me or not me.baziness_user_id:
        raise HTTPException(409, "call /api/platform/token first")
    return {"linkCode": platform.create_page_link_code(me.baziness_user_id)}

@app.post("/api/platform/login")
def platform_login(body: LoginRequest):
    if not platform_enabled():
        return {"ok": False, "reason": "disabled"}
    try:
        baziness_user_id = platform.validate_user_access_token(body.token) # POST now;
        verify right away
    except Exception:
        return {"ok": False, "reason": "invalid"}
    owner = db.find_user_by_baziness_user_id(baziness_user_id)
    me = db.ensure_user_by_device(body.deviceId)
    if owner and owner.id != me.id:
        return {"ok": True, "switchToDeviceId": owner.device_id}
    if me.baziness_user_id != baziness_user_id:
        db.set_refresh_token(me.id, None) # belonged to
        the old user
    db.link_baziness(me.id, baziness_user_id, True)
    return {"ok": True} # the client
    then calls /token again

@app.post("/api/platform/logout") # the page sent `logout`
def platform_logout(body: DeviceRequest):
    me = db.find_user_by_device(body.deviceId)
    if me and me.baziness_refresh_token:
        platform.end_user_session(me.baziness_refresh_token)
        db.set_refresh_token(me.id, None)
    return {"ok": True}

```

Client, step by step

1. **On boot**, send `linkCode` (from `?linkCode=` in the address, or from your deep link) to `/api/platform/token` instead of `linkedToken`. Remove both `linkCode` and `linkedToken` from the address afterwards: a code works once.
2. **Stop storing the token** for later boots (`localStorage`, `PlayerPrefs`). Your server keeps the session and gives you a fresh token on every `/api/platform/token`.
3. **Before using the token** (`user/get-coin`), check `tokenExpiresAt`. If it expires within a minute, or a call answers `401`, call `/api/platform/token` again.
4. **To open a Baziness page**, get a code from `/api/platform/link-code` and put `linkCode=` `<code>` in the page address instead of `linkedToken`. All other parameters stay the same (pages).
5. **On the page's logout message**, call `/api/platform/logout`, then start over as before.

TypeScript

```
// PlatformController – the parts that change.

// 1. Boot: forward the one-time ?linkCode= (it replaces ?linkedToken=). Don't keep the token in
// localStorage any more: your server keeps the session and gives you a fresh token on demand.
async init(): Promise<boolean> {
  const params = new URLSearchParams(location.search);
  const linkCode = params.get("linkCode");
  const res = await fetch("/api/platform/token", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ deviceId: this.deviceId, linkCode }),
  });
  const data = await res.json();
  if (data.switchToDeviceId) return this.switchTo(data.switchToDeviceId); // unchanged
  this.token = data.token ?? null;
  this.tokenExpiresAt = data.tokenExpiresAt ? Date.parse(data.tokenExpiresAt) : 0;
  if (linkCode || params.has("linkedToken")) { // a code works once
    params.delete("linkCode"); params.delete("linkedToken");
    const rest = params.toString();
    history.replaceState(null, "", rest ? `${location.pathname}?${rest}` : location.pathname);
  }
  return !!this.token;
}

// 2. The token lives 15 minutes: ask your server again shortly before it expires, or after a 401.
async accessToken(): Promise<string | null> {
  if (!this.token || this.tokenExpiresAt - Date.now() < 60_000) await this.init();
  return this.token;
}

async getCoin(): Promise<number> {
  const call = async () => fetch(`${this.apiBase}/user/get-coin`, {
    headers: { Authorization: `Bearer ${await this.accessToken()}` },
  });
```

```

});
let res = await call();
if (res.status === 401) { this.token = null; res = await call(); } // renew once,
then give up
return res.ok ? Number((await res.json()).amount ?? 0) : 0;
}

// 3. Pages: a fresh one-time code for every page you open, instead of the token.
async pageUrl(path: string, o: PageOptions = {}): Promise<string> {
    const res = await fetch("/api/platform/link-code", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId }),
    });
    const { linkCode } = await res.json();
    const url = new URL(bazinessPageUrl(this.webBase, path, this.packageName, null, o)); // no
    token any more
    url.searchParams.set("linkCode", linkCode);
    return url.toString();
}

// 4. The page sent `logout`: tell your server first, then start over as before.
async onPageLogout() {
    await fetch("/api/platform/logout", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId }),
    }).catch(() => {});
    this.resetAndRestart(); // unchanged: delete local data, new visitor
}

```

C# (Unity)

```

// BazinessLogin – the parts that change.
public DateTimeOffset TokenExpiresAt { get; private set; }

[Serializable] class LinkCodeResponse { public string linkCode; }

// 1. Boot: forward the one-time ?linkCode= / deep-link code (it replaces linkedToken).
// Stop saving the token in PlayerPrefs: your server keeps the session.
public IEnumerator Init(string deviceId, string linkCode = null)
{
    var body = $"{{\"deviceId\":\"{deviceId}\",\"linkCode\":{Quote(linkCode)}}}";
    using var req = Post("/api/platform/token", body);
    yield return req.SendWebRequest();
    var r = JsonUtility.FromJson<TokenResponse>(req.downloadHandler.text); // TokenResponse
    gains: string tokenExpiresAt
    if (!string.IsNullOrEmpty(r.switchToDeviceId)) { SwitchTo(r.switchToDeviceId); yield
    break; } // unchanged
    Token = r.token;
    TokenExpiresAt = DateTimeOffset.TryParse(r.tokenExpiresAt, out var t) ? t :
    DateTimeOffset.MinValue;
}

// 2. The token lives 15 minutes: renew it through your server before using it.
public IEnumerator EnsureFreshToken(string deviceId)
{
    if (string.IsNullOrEmpty(Token) || TokenExpiresAt - DateTimeOffset.UtcNow <
    TimeSpan.FromMinutes(1))

```

```

        yield return Init(deviceId);
    }

    // 3. Pages: a fresh one-time code for every page you open, instead of the token.
    public IEnumerator OpenPage(string deviceId, string path, Action<string> open,
        string redirect = null, bool allowGuest = true, string extra = null)
    {
        using var req = Post("/api/platform/link-code", $"{{"deviceId\":\"{deviceId}\"}}");
        yield return req.SendWebRequest();
        var linkCode = JsonUtility.FromJson<LinkCodeResponse>(req.downloadHandler.text).linkCode;
        open(PageUrl(path, linkCode, redirect, allowGuest, extra));
    }

    public string PageUrl(string path, string linkCode, string redirect = null, bool allowGuest =
    true, string extra = null)
    {
        var url = $"{webBase.TrimEnd('/')}{path}?packageName=
        {Uri.EscapeDataString(packageName)}&sdkType=js";
        if (!string.IsNullOrEmpty(linkCode)) url += $"&linkCode=
        {Uri.EscapeDataString(linkCode)}"; // was linkedToken
        if (!string.IsNullOrEmpty(redirect)) url += $"&redirect={redirect}";
        if (!allowGuest) url += "&allowGuest=false";
        if (!string.IsNullOrEmpty(extra)) url += $"&{extra}";
        return url;
    }
}

```

The new platform endpoints

All are under the usual `{api}` base. The `client` endpoints need your **client token**, as before.

Create a user, with a session

HTTP

```

POST {api}/v2/client/create-user
Authorization: Bearer <client token>
Content-Type: application/json

{ "deviceId": "8f1c2a...", "platform": "Android", "store": "CafeBazaar", "linkCode": null }

→ 200 OK
{
    "userId": 239236,
    "token": "eyJhbGciOiJIUzI1NiIs...", // the access token (same as
tokens.accessToken)
    "isLinkedToAccount": false,
    "tokens": {
        "accessToken": "eyJhbGciOiJIUzI1NiIs...", // 15 minutes – the client uses this
        "accessTokenExpiresAt": "2026-10-22T10:15:00+00:00",
        "refreshToken": "rt1.dkUoqaDDG8YB15KAQN01q77FE...", // keep on YOUR server only
        "refreshTokenExpiresAt": "2026-11-21T10:00:00+00:00"
    }
}

```

```
→ 400 { "success": false, "error": "invalid_link_code" } // the code expired or was already used
```

Renew a session

No client token: the refresh token itself is the credential. Call it **from your server**.

HTTP

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

{ "refreshToken": "rt1.dkUoqaDDG8YB15KAQN01q77FE..." }

→ 200 OK
{
  "userId": 239236,
  "isLinkedToAccount": true,
  "accessToken": "eyJhbGciOiJIUzI1NiIs...",
  "accessTokenExpiresAt": "2026-10-22T10:30:00+00:00",
  "refreshToken": "rt1.Wn0QhZ...", // store THIS one – the old one no longer works
  "refreshTokenExpiresAt": "2026-11-21T10:15:00+00:00"
}

→ 401 { "success": false, "error": "refresh_token_rotated" } // another request refreshed first: reload yours, retry once
→ 401 { "success": false, "error": "session_revoked" } // also invalid_refresh_token, refresh_token_expired, // refresh_token_reused: get a new session
→ 403 { "success": false, "error": "account_suspended" }
```

A new session for a known player

HTTP

```
POST {api}/v2/client/get-user-token/239236
Authorization: Bearer <client token>

→ 200 OK
{ "userId": 239236, "token": "eyJhbGciOiJIUzI1NiIs...", "isLinkedToAccount": true,
  "tokens": { "accessToken": "...", "accessTokenExpiresAt": "...", "refreshToken": "rt1...", "refreshTokenExpiresAt": "..." } }

→ 403 { "success": false, "error": "access denied" } // the user belongs to another game
```

A link code for opening a page

HTTP

```
POST {api}/v2/client/link-code/239236
```

```
Authorization: Bearer <client token>
```

```
→ 200 OK
```

```
{ "code": "q3Jc9yX2tL0...", "expiresAt": "2026-10-22T10:02:00+00:00" } // one page, once, ~2 minutes
```

```
→ 403 { "success": false, "error": "access denied" } // the user belongs to another game
```

Verify a token (body instead of address)

HTTP

```
POST {api}/client/validate-user-token
```

```
Authorization: Bearer <client token>
```

```
Content-Type: application/json
```

```
{ "token": "eyJhbGciOiJIUzI1NiIs..." }
```

```
→ 200 OK
```

```
{ "userId": 239236, "isValid": true }
```

```
→ 200 OK (invalid or expired: not an HTTP error)
```

```
{ "userId": 0, "isValid": false }
```

End a session

`POST {api}/auth/logout` with `{ "refreshToken": "rt1..." }`. It always answers `200`.

Errors

Answer	Where	Meaning	What to do
401 refresh_token_rotated	auth/refresh	a parallel request of this player refreshed first	reload the saved refresh token, retry once
401 session_revoked , refresh_token_expired , invalid_refresh_token , refresh_token_reused	auth/refresh	the session is over	v2/client/get-user-token/{id} , save the new refresh token
403 account_suspended	auth/refresh	the account is blocked	show the blocked message
400 invalid_link_code	v2/client/create-user	the code expired, was used, or wasn't meant for your game	continue without the code
403	v2/client/link-code/{id} , v2/client/get-user-token/{id}	the user belongs to another game	check the userId you saved
401	user/get-coin (client)	the 15-minute token expired	ask /api/platform/token again, retry once

Checklist

Test on the [development platform](#) (any phone number, code 111111):

- ☐ A new player boots: the client gets a token; your database has a refresh token for them.
- ☐ Boot again: `auth/refresh` is used (not `get-user-token`), and the stored refresh token **changed**.

- ☐ Two boots of the same player at the same moment: both succeed (one through the `rotated` retry).
- ☐ An existing player from before the migration boots once and gets a refresh token.
- ☐ Every Baziness page opens with `linkCode=...` and **no** `linkedToken` ; the sign-in, profile, reward shop and offer pages all work.
- ☐ Signing in on a page with a phone that already belongs to another of your players still switches to that player.
- ☐ Opening your game from the Baziness app with `?linkCode=` lands in the right account; open the same link twice → the second time the code is ignored and the game still boots.
- ☐ After 15 minutes of play, the coin balance still loads (token renewed through your server).
- ☐ `logout` from a page ends the session; the next boot is a new visitor.
- ☐ Your logs contain no refresh tokens, and no tokens in addresses.

Note

Until the cut-off date, the Baziness app also sends `?linkedToken=` . Once your game reads `linkCode` , ignore `linkedToken` . After the cut-off it won't be sent any more, and old tokens stop working.

Server code

Ready-made server code for every step — the platform client, your own endpoints, Gita coin payouts, Baziness Pay and campaign tags — in TypeScript, C# and Python.

These samples implement the [API steps](#) and [guides](#). Each one comes in **TypeScript (Node)**, **C# (.NET)** and **Python**; pick a tab, and the choice is remembered on every page. They use Express, ASP.NET minimal APIs and FastAPI, but the logic carries over to any framework. Where a sample calls `db...`, plug in your own database.

Settings

.env (server)

```
# --- Baziness platform: accounts, Gita coins, reward pages -----
BAZINESS_PLATFORM_ENABLED=false           # master switch: nothing shows until "true"
BAZINESS_PLATFORM_USE_PRODUCTION=false    # selects the default API + web URLs below
BAZINESS_CLIENT_ID=
BAZINESS_CLIENT_SECRET=
BAZINESS_PLATFORM_PACKAGE_NAME=com.yourstudio.yourgame
# Optional overrides (derived from USE_PRODUCTION when blank):
# dev api https://platform-dev.karizmastudios.org/api    web https://platform-web-dev.karizmastudios.org
# prod api https://platform.karizmastudio.com/api        web https://baziness.com
BAZINESS_PLATFORM_API_URL=
BAZINESS_PLATFORM_WEB_URL=
BAZINESS_GAME_NAME=yourgame               # ?game_name= of your pages on the platform
(rewards, game offer)
# Optional: the tutorial offer page (blank = {web}/game-offer?game_name=<BAZINESS_GAME_NAME>)
BAZINESS_GAME_OFFER_URL=

# --- Baziness Pay: real-money purchases -----
BAZINESS_PAY_ENABLED=false                # buy buttons stay disabled until "true"
# dev https://pay-dev.karizmastudios.org    prod https://pay.baziness.com
BAZINESS_PAY_BASE_URL=https://pay-dev.karizmastudios.org
BAZINESS_PAY_PACKAGE_NAME=com.yourstudio.yourgame # the Pay app's package (pname) – may
differ from BAZINESS_PLATFORM_PACKAGE_NAME
BAZINESS_SKU_PREFIX=                      # optional prefix: SKU = prefix + product id
DEV_FAKE_BAZINESS_PAY=false               # local development only (see the Pay page)
```

The platform client

One small class that does [steps 1–5](#): it gets the client token on first use, gets it again after a `401`, and builds payout ids from a grant key.

TypeScript (Node)

```
// platform/baziness.ts - all credentials stay on the server.
import { createHash } from "node:crypto";

const DEV_API = "https://platform-dev.karizmastudios.org/api";
const PROD_API = "https://platform.karizmastudio.com/api";
const DEV_WEB = "https://platform-web-dev.karizmastudios.org";
const PROD_WEB = "https://baziness.com";

const env = process.env;
const ENABLED = env.BAZINESS_PLATFORM_ENABLED === "true";
const USE_PROD = env.BAZINESS_PLATFORM_USE_PRODUCTION === "true";
const CLIENT_ID = env.BAZINESS_CLIENT_ID ?? "";
const CLIENT_SECRET = env.BAZINESS_CLIENT_SECRET ?? "";
const PACKAGE_NAME = env.BAZINESS_PLATFORM_PACKAGE_NAME ?? "";
const trim = (u: string) => u.replace(/\/+$/, "");
export const API_BASE = trim(env.BAZINESS_PLATFORM_API_URL || (USE_PROD ? PROD_API : DEV_API));
export const WEB_BASE = trim(env.BAZINESS_PLATFORM_WEB_URL || (USE_PROD ? PROD_WEB : DEV_WEB));

export const platformEnabled = () => ENABLED && CLIENT_ID !== "" && CLIENT_SECRET !== "";

/** Public, non-secret config for the browser. */
export const platformClientConfig = () => ({ webBase: WEB_BASE, apiBase: API_BASE,
packageName: PACKAGE_NAME });

export interface PlatformUser { userId: number; token: string; isLinkedToAccount: boolean }

let bearer: string | null = null;

async function authenticate(): Promise<void> {
  const res = await fetch(`${API_BASE}/client/auth`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ clientId: CLIENT_ID, clientSecret: CLIENT_SECRET }),
  });
  if (!res.ok) throw new Error(`client/auth failed (${res.status})`);
  const json = (await res.json()) as Record<string, unknown>;
  bearer = String(json.token ?? json.Token ?? "");
  if (!bearer) throw new Error("client/auth returned no token");
}

async function platformFetch(method: "GET" | "POST", path: string, body?: unknown) {
  if (!bearer) await authenticate();
  const send = () =>
    fetch(`${API_BASE}${path}`, {
      method,
      headers: { Authorization: `Bearer ${bearer}`, ...(body ? { "Content-Type":
"application/json" } : {}) },
      body: body ? JSON.stringify(body) : undefined,
    });
  let res = await send();
  if (res.status === 401) { await authenticate(); res = await send(); } // bearer expired
  if (!res.ok) throw Object.assign(new Error(`${path} failed (${res.status})`), { status:
res.status }); // 400 from add-coin = already paid
  return (await res.json()) as Record<string, unknown>;
}
```

```

const readUser = (j: Record<string, unknown>): PlatformUser => ({
  userId: Number(j.userId ?? j.UserId ?? 0),
  token: String(j.token ?? j.Token ?? ""),
  isLinkedToAccount: Boolean(j.isLinkedToAccount ?? j.IsLinkedToAccount ?? false),
});

export const createUser = async (deviceId: string, platform: string, store: string,
linkedToken: string | null) =>
  readUser(await platformFetch("POST", "/client/create-user", { deviceId, platform, store,
linkedToken }));

export const getUserAccessToken = async (baziinessUserId: number) =>
  readUser(await platformFetch("GET", `/client/get-user-token/${baziinessUserId}`));

export async function validateUserAccessToken(token: string): Promise<number> {
  const j = await platformFetch("GET", `/client/validate-user-
token/${encodeURIComponent(token)}`);
  if (!(j.isValid ?? j.IsValid)) throw new Error("invalid user token");
  return Number(j.userId ?? j.UserId ?? 0);
}

/** Credit Gita coins. `grantKey` makes the transaction id deterministic: a repeat answers
400 (= already paid). */
export async function addCoin(baziinessUserId: number, amount: number, grantKey: string,
metaData?: object) {
  const hash = createHash("sha1").update(grantKey).digest("hex").slice(0, 24);
  const j = await platformFetch("POST", "/client/add-coin", {
    userId: baziinessUserId, amount, transactionId: `${baziinessUserId}_${amount}_${hash}`,
    metaData: metaData ?? null,
  });
  return Number(j.amount ?? j.Amount ?? 0); // the player's new balance
}

```

C# (.NET)

```

// BazinessPlatformClient.cs - all credentials stay on the server.
// Program.cs: builder.Services.AddSingleton<PlatformSettings>(...);
// builder.Services.AddHttpClient<BazinessPlatformClient>();
using System.Net;
using System.Net.Http.Headers;
using System.Net.Http.Json;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;

public record PlatformSettings(bool Enabled, bool UseProduction, long ClientId, string
ClientSecret, string PackageName)
{
  public string ApiBase => UseProduction ? "https://platform.karizmastudio.com/api" :
"https://platform-dev.karizmastudios.org/api";
  public string WebBase => UseProduction ? "https://baziiness.com" : "https://platform-web-
dev.karizmastudios.org";
  public bool IsOn => Enabled && ClientId != 0 && !string.IsNullOrEmpty(ClientSecret);
}

public record PlatformUser(long UserId, string Token, bool IsLinkedToAccount);

public sealed class BazinessPlatformClient(HttpClient http, PlatformSettings settings)

```

```

{
    static readonly JsonSerializerOptions Json = new(JsonSerializerDefaults.Web); //
    camelCase + case-insensitive
    static string? bearer; // the client token, shared by
    every request
    static readonly SemaphoreSlim authLock = new(1, 1);

    async Task AuthenticateAsync()
    {
        await authLock.WaitAsync();
        try
        {
            var res = await http.PostAsJsonAsync($"{settings.ApiBase}/client/auth",
                new { clientId = settings.ClientId, clientSecret = settings.ClientSecret });
            res.EnsureSuccessStatusCode();
            bearer = (await res.Content.ReadFromJsonAsync<JsonElement>
                ()).GetProperty("token").GetString()
                ?? throw new InvalidOperationException("client/auth returned no token");
        }
        finally { authLock.Release(); }
    }

    async Task<JsonElement> SendAsync(HttpMethod method, string path, object? body = null)
    {
        if (bearer is null) await AuthenticateAsync();
        async Task<HttpResponseMessage> Send()
        {
            var req = new HttpRequestMessage(method, $"{settings.ApiBase}{path}");
            req.Headers.Authorization = new AuthenticationHeaderValue("Bearer", bearer);
            if (body is not null) req.Content = JsonContent.Create(body, options: Json);
            return await http.SendAsync(req);
        }
        var res = await Send();
        if (res.StatusCode == HttpStatusCode.Unauthorized) { await AuthenticateAsync(); res =
        await Send(); } // token expired
        if (!res.IsSuccessStatusCode) throw new HttpRequestException($"{path} failed
        ({(int)res.StatusCode})", null, res.StatusCode);
        return await res.Content.ReadFromJsonAsync<JsonElement>(Json);
    }

    static PlatformUser ReadUser(JsonElement j) => new(
        j.GetProperty("userId").GetInt64(), j.GetProperty("token").GetString()!,
        j.GetProperty("isLinkedToAccount").GetBoolean());

    public async Task<PlatformUser> CreateUserAsync(string deviceId, string platform, string
    store, string? linkedToken) =>
        ReadUser(await SendAsync(HttpMethod.Post, "/client/create-user", new { deviceId,
        platform, store, linkedToken }));

    public async Task<PlatformUser> GetUserAccessTokenAsync(long businessUserId) =>
        ReadUser(await SendAsync(HttpMethod.Get, $"/client/get-user-
        token/{businessUserId}"));

    /// <summary>The server-side gate for every token a client sends. Throws when the token
    is invalid.</summary>
    public async Task<long> ValidateUserAccessTokenAsync(string token)
    {
        var j = await SendAsync(HttpMethod.Get, $"/client/validate-user-
        token/{Uri.EscapeDataString(token)}");
        if (!j.GetProperty("isValid").GetBoolean()) throw new

```

```

InvalidOperationException("invalid user token");
        return j.GetProperty("userId").GetInt64();
    }

    /// <summary>
    /// Credit Gita coins. The transaction id comes from grantKey (what the payout IS, e.g.
    "match:17"),
    /// so a repeat is refused with HTTP 400 – treat that 400 as "already paid".
    /// </summary>
    public async Task<long> AddCoinAsync(long bazinessUserId, long amount, string grantKey,
    object? metaData = null)
    {
        var hash = Convert.ToHexString(SHA1.HashData(Encoding.UTF8.GetBytes(grantKey)))
[..<24].ToLowerInvariant();
        var j = await SendAsync(HttpMethod.Post, "/client/add-coin",
            new { userId = bazinessUserId, amount, transactionId = $"
{bazinessUserId}_{amount}_{hash}", metaData });
        return j.GetProperty("amount").GetInt64(); // the player's new balance
    }
}

```

Python

```

# baziness/platform.py – all credentials stay on the server.
import hashlib
import os
import requests

DEV_API, PROD_API = "https://platform-dev.karizmastudios.org/api",
"https://platform.karizmastudio.com/api"
DEV_WEB, PROD_WEB = "https://platform-web-dev.karizmastudios.org", "https://baziness.com"

ENABLED = os.getenv("BAZINESS_PLATFORM_ENABLED", "false").lower() == "true"
USE_PROD = os.getenv("BAZINESS_PLATFORM_USE_PRODUCTION", "false").lower() == "true"
CLIENT_ID = os.getenv("BAZINESS_CLIENT_ID", "")
CLIENT_SECRET = os.getenv("BAZINESS_CLIENT_SECRET", "")
PACKAGE_NAME = os.getenv("BAZINESS_PLATFORM_PACKAGE_NAME", "")
API_BASE = (os.getenv("BAZINESS_PLATFORM_API_URL") or (PROD_API if USE_PROD else
DEV_API)).rstrip("/")
WEB_BASE = (os.getenv("BAZINESS_PLATFORM_WEB_URL") or (PROD_WEB if USE_PROD else
DEV_WEB)).rstrip("/")

def platform_enabled() -> bool:
    return ENABLED and bool(CLIENT_ID) and bool(CLIENT_SECRET)

def platform_client_config() -> dict:
    """Public, non-secret config for the browser."""
    return {"webBase": WEB_BASE, "apiBase": API_BASE, "packageName": PACKAGE_NAME}

def _pick(j: dict, name: str, default=None):
    # The platform is .NET: accept camelCase and PascalCase.
    return j.get(name, j.get(name[0].upper() + name[1:], default))

class BazinessPlatform:
    def __init__(self) -> None:

```

```

self._bearer: str | None = None
self._http = requests.Session()

def _authenticate(self) -> None:
    r = self._http.post(f"{API_BASE}/client/auth",
                        json={"clientId": int(CLIENT_ID), "clientSecret": CLIENT_SECRET},
                        timeout=10)
    r.raise_for_status()
    self._bearer = _pick(r.json(), "token")
    if not self._bearer:
        raise RuntimeError("client/auth returned no token")

def _call(self, method: str, path: str, body: dict | None = None) -> dict:
    if not self._bearer:
        self._authenticate()
    send = lambda: self._http.request(method, f"{API_BASE}{path}", json=body, timeout=10,
                                       headers={"Authorization": f"Bearer {self._bearer}"})
    r = send()
    if r.status_code == 401: # bearer expired
        self._authenticate()
        r = send()
    r.raise_for_status()
    return r.json()

@staticmethod
def _user(j: dict) -> dict:
    return {"userId": int(_pick(j, "userId", 0)), "token": str(_pick(j, "token", "")),
            "isLinkedToAccount": bool(_pick(j, "isLinkedToAccount", False))}

def create_user(self, device_id: str, platform: str, store: str, linked_token: str | None) -> dict:
    return self._user(self._call("POST", "/client/create-user", {
        "deviceId": device_id, "platform": platform, "store": store, "linkedToken": linked_token}))

def get_user_access_token(self, business_user_id: int) -> dict:
    return self._user(self._call("GET", f"/client/get-user-token/{business_user_id}"))

def validate_user_access_token(self, token: str) -> int:
    j = self._call("GET", f"/client/validate-user-token/{requests.utils.quote(token, safe='')}")
    if not _pick(j, "isValid", False):
        raise ValueError("invalid user token")
    return int(_pick(j, "userId", 0))

def add_coin(self, business_user_id: int, amount: int, grant_key: str, meta: dict | None = None) -> int:
    digest = hashlib.sha1(grant_key.encode()).hexdigest()[:24]
    j = self._call("POST", "/client/add-coin", {
        "userId": business_user_id, "amount": amount,
        "transactionId": f"{business_user_id}_{amount}_{digest}", "metaData": meta})
    return int(_pick(j, "amount", 0)) # the player's new balance

platform = BazinessPlatform()

```

Telling the client the settings

`GET /api/platform/config`: whether Baziness is on, and where its pages and API are. It has no side effects and never includes the secret.

TypeScript (Node)

```
import express from "express";
import { platformClientConfig, platformEnabled, WEB_BASE } from "../platform/baziness";

const GAME_NAME = process.env.BAZINESS_GAME_NAME ?? "example"; // your game's name on the platform
const GAME_OFFER_URL = process.env.BAZINESS_GAME_OFFER_URL || `${WEB_BASE}/game-offer?game_name=${encodeURIComponent(GAME_NAME)}`;

const app = express();
app.use(express.json());

// Side-effect free: tells the client whether Baziness is on and where its pages live. Never the secret.
app.get("/api/platform/config", (_req, res) => {
  res.json(platformEnabled()
    ? { enabled: true, ...platformClientConfig(), gameId: GAME_NAME, gameOfferUrl: GAME_OFFER_URL }
    : { enabled: false });
});
```

C# (.NET)

```
// Side-effect free: tells the client whether Baziness is on and where its pages live. Never the secret.
app.MapGet("/api/platform/config", (PlatformSettings s, IConfiguration cfg) =>
{
  if (!s.IsOn) return Results.Ok(new { enabled = false });
  var gameId = cfg["Baziness:GameName"] ?? "example";
  var gameOfferUrl = cfg["Baziness:GameOfferUrl"] is { Length: > 0 } custom
    ? custom
    : $"{s.WebBase}/game-offer?game_name={Uri.EscapeDataString(gameId)}";
  return Results.Ok(new { enabled = true, webBase = s.WebBase, apiBase = s.ApiBase,
    packageName = s.PackageName, gameId, gameOfferUrl });
});
```

Python

```
import os
from urllib.parse import quote
from fastapi import FastAPI
from baziness.platform import WEB_BASE, platform_client_config, platform_enabled

GAME_NAME = os.getenv("BAZINESS_GAME_NAME", "example") # your game's name on the platform
GAME_OFFER_URL = os.getenv("BAZINESS_GAME_OFFER_URL") or f"{WEB_BASE}/game-offer?game_name={quote(GAME_NAME)}"

app = FastAPI()

@app.get("/api/platform/config")
```

```
def platform_config():
    """Side-effect free: tells the client whether Baziness is on and where its pages live.
    Never the secret."""
    if not platform_enabled():
        return {"enabled": False}
    return {"enabled": True, **platform_client_config(), "gameName": GAME_NAME,
"gameOfferUrl": GAME_OFFER_URL}
```

Login endpoints

`POST /api/platform/token` and `POST /api/platform/login`, as described in [Login → what your server provides](#).

TypeScript (Node)

```
import { createUser, getUserAccessToken, platformClientConfig, platformEnabled,
validateUserAccessToken } from "../platform/baziness";
import { db } from "../db"; // your own data layer

app.post("/api/platform/token", async (req, res) => {
    const deviceId = String(req.body.deviceId ?? "").trim();
    const linkedToken: string | null = req.body.linkedToken ?? null;
    if (!deviceId) return res.status(400).json({ error: "deviceId required" });
    if (!platformEnabled()) return res.json({ enabled: false });

    const config = { enabled: true, ...platformClientConfig() };
    // Pass first-touch attribution so a player who signs in before your normal login is still
    credited.
    const user = await db.ensureUserByDevice(deviceId, { utm: req.body.utm, tag: req.body.tag
});
    try {
        if (!user.bazinessUserId) {
            const created = await createUser(deviceId, "Web", "Direct", linkedToken);
            const owner = await db.findUserByBazinessId(created.userId);
            if (owner && owner.id !== user.id) {
                return res.json({ ...config, token: created.token, bazinessUserId: created.userId,
isLinkedToAccount: created.isLinkedToAccount, switchToDeviceId: owner.deviceId });
            }
            await db.linkBaziness(user.id, created.userId, created.isLinkedToAccount);
            return res.json({ ...config, token: created.token, bazinessUserId: created.userId,
isLinkedToAccount: created.isLinkedToAccount });
        }
        const fresh = await getUserAccessToken(user.bazinessUserId);
        if (fresh.isLinkedToAccount !== user.isBazinessLinked) await db.setLinked(user.id,
fresh.isLinkedToAccount);
        return res.json({ ...config, token: fresh.token, bazinessUserId: fresh.userId,
isLinkedToAccount: fresh.isLinkedToAccount });
    } catch (err) {
        console.error("[platform] token failed:", (err as Error).message);
        return res.json({ ...config, error: "platform_unavailable" }); // never block the boot
    }
});

app.post("/api/platform/login", async (req, res) => {
    const deviceId = String(req.body.deviceId ?? "").trim();
    const token = String(req.body.token ?? "").trim();
```

```

    if (!deviceId || !token) return res.status(400).json({ error: "deviceId and token required"
});
    if (!platformEnabled()) return res.json({ ok: false, reason: "disabled" });
    try {
        const bazinessUserId = await validateUserAccessToken(token); // the server-side gate
        const owner = await db.findUserByBazinessId(bazinessUserId);
        const me = await db.ensureUserByDevice(deviceId);
        if (owner && owner.id !== me.id) return res.json({ ok: true, switchToDeviceId:
owner.deviceId });
        await db.linkBaziness(me.id, bazinessUserId, true);
        return res.json({ ok: true });
    } catch {
        return res.json({ ok: false, reason: "invalid" });
    }
});

```

C# (.NET)

```

app.MapPost("/api/platform/token", async (TokenRequest body, BazinessPlatformClient platform,
    IUserStore users, PlatformSettings pub, ILogger<Program> log) =>
{
    if (string.IsNullOrEmpty(body.DeviceId)) return Results.BadRequest(new { error =
"deviceId required" });
    if (!pub.IsOn) return Results.Ok(new { enabled = false });

    var user = await users.EnsureByDeviceAsync(body.DeviceId);
    try
    {
        PlatformUser auth;
        if (user.BazinessUserId is null)
        {
            auth = await platform.CreateUserAsync(body.DeviceId, "Web", "Direct",
body.LinkedToken);
            var owner = await users.FindByBazinessUserIdAsync(auth.UserId);
            if (owner is not null && owner.Id != user.Id)
                return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase,
pub.PackageName,
                    token = auth.Token, bazinessUserId = auth.UserId, auth.IsLinkedToAccount,
                    switchToDeviceId = owner.DeviceId });
            await users.LinkBazinessAsync(user.Id, auth.UserId, auth.IsLinkedToAccount);
        }
        else
        {
            auth = await platform.GetUserAccessTokenAsync(user.BazinessUserId.Value);
            if (auth.IsLinkedToAccount != user.IsBazinessLinked)
                await users.SetLinkedAsync(user.Id, auth.IsLinkedToAccount);
        }
        return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
            token = auth.Token, bazinessUserId = auth.UserId, auth.IsLinkedToAccount });
    }
    catch (Exception ex)
    {
        log.LogError(ex, "platform token failed");
        return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
            error = "platform_unavailable" }); // never block the boot
    }
});

app.MapPost("/api/platform/login", async (LoginRequest body, BazinessPlatformClient platform,

```

```

IUserStore users, PlatformSettings pub) =>
{
    if (string.IsNullOrEmpty(body.DeviceId) || string.IsNullOrEmpty(body.Token))
        return Results.BadRequest(new { error = "deviceId and token required" });
    if (!pub.IsOn) return Results.Ok(new { ok = false, reason = "disabled" });
    try
    {
        long bazinessUserId = await platform.ValidateUserAccessTokenAsync(body.Token); //
        throws when invalid
        var owner = await users.FindByBazinessUserIdAsync(bazinessUserId);
        var me = await users.EnsureByDeviceAsync(body.DeviceId);
        if (owner is not null && owner.Id != me.Id)
            return Results.Ok(new { ok = true, switchToDeviceId = owner.DeviceId });
        await users.LinkBazinessAsync(me.Id, bazinessUserId, isLinked: true);
        return Results.Ok(new { ok = true });
    }
    catch
    {
        return Results.Ok(new { ok = false, reason = "invalid" });
    }
});

public record TokenRequest(string DeviceId, string? LinkedToken);
public record LoginRequest(string DeviceId, string Token);

```

Python

```

from fastapi import FastAPI
from pydantic import BaseModel
from baziness.platform import platform, platform_client_config, platform_enabled
from app import db # your own data layer

app = FastAPI()

class TokenRequest(BaseModel):
    deviceId: str
    linkedToken: str | None = None

class LoginRequest(BaseModel):
    deviceId: str
    token: str

@app.post("/api/platform/token")
def platform_token(body: TokenRequest):
    if not platform_enabled():
        return {"enabled": False}
    config = {"enabled": True, **platform_client_config()}
    user = db.ensure_user_by_device(body.deviceId)
    try:
        if not user.baziness_user_id:
            created = platform.create_user(body.deviceId, "Web", "Direct", body.linkedToken)
            owner = db.find_user_by_baziness_user_id(created["userId"])
            if owner and owner.id != user.id:
                return {**config, "token": created["token"], "bazinessUserId":
created["userId"],
                        "isLinkedToAccount": created["isLinkedToAccount"],

```

```

"switchToDeviceId": owner.device_id}
    db.link_baziness(user.id, created["userId"], created["isLinkedToAccount"])
    auth = created
  else:
    auth = platform.get_user_access_token(user.baziness_user_id)
    if auth["isLinkedToAccount"] != user.is_baziness_linked:
      db.set_linked(user.id, auth["isLinkedToAccount"])
    return {**config, "token": auth["token"], "bazinessUserId": auth["userId"],
            "isLinkedToAccount": auth["isLinkedToAccount"]}
except Exception as exc: # platform down: never block the boot
  print("[platform] token failed:", exc)
  return {**config, "error": "platform_unavailable"}

@app.post("/api/platform/login")
def platform_login(body: LoginRequest):
  if not platform_enabled():
    return {"ok": False, "reason": "disabled"}
  try:
    baziness_user_id = platform.validate_user_access_token(body.token) # the server-side
gate
  except Exception:
    return {"ok": False, "reason": "invalid"}
  owner = db.find_user_by_baziness_user_id(baziness_user_id)
  me = db.ensure_user_by_device(body.deviceId)
  if owner and owner.id != me.id:
    return {"ok": True, "switchToDeviceId": owner.device_id}
  db.link_baziness(me.id, baziness_user_id, True)
  return {"ok": True}

```

Giving Gita coins

A payout that can never be paid twice: the grant key fixes the `transactionId`, and a `400` means it was already paid.

TypeScript (Node)

```

// grants.ts - a payout that can never be paid twice.
import { addCoin } from "../platform/baziness";

export async function payTournamentReward(claimId: number, bazinessUserId: number, amount:
number) {
  try {
    // Same claim → same key → same transactionId → the platform refuses a second credit.
    await addCoin(bazinessUserId, amount, `reward_claim:${claimId}`, { source: "tournament",
claimId });
  } catch (err) {
    // 400 = this transactionId was already credited (e.g. we crashed after the platform paid
but
    // before we saved): the coins are there, so carry on instead of failing forever.
    if ((err as { status?: number }).status !== 400) throw err;
  }
  await db.claims.markPaid(claimId); // your own bookkeeping
  await db.ledger.insert({ claimId, currency: "GITA_COIN", amount, external: true });
}

```

C# (.NET)

```
// Grants.cs - a payout that can never be paid twice.
public sealed class Grants(BazinessPlatformClient platform, AppDb db)
{
    public async Task PayTournamentRewardAsync(long claimId, long bazinessUserId, long
amount)
    {
        try
        {
            // Same claim → same key → same transactionId → the platform refuses a second
            credit.
            await platform.AddCoinAsync(bazinessUserId, amount, $"reward_claim:{claimId}",
new { source = "tournament", claimId });
        }
        catch (HttpRequestException e) when (e.StatusCode ==
System.Net.HttpStatusCode.BadRequest)
        {
            // 400 = already credited by an earlier attempt: the coins are there, carry on.
        }
        await db.Claims.MarkPaidAsync(claimId); // your own bookkeeping
        await db.Ledger.InsertAsync(new LedgerRow(claimId, "GITA_COIN", amount, External:
true));
    }
}
```

Python

```
# grants.py - a payout that can never be paid twice.
import requests
from baziness.platform import platform

def pay_tournament_reward(claim_id: int, baziness_user_id: int, amount: int) -> None:
    try:
        # Same claim → same key → same transactionId → the platform refuses a second credit.
        platform.add_coin(baziness_user_id, amount, f"reward_claim:{claim_id}", {"source":
"tournament", "claimId": claim_id})
    except requests.HTTPError as err:
        if err.response is None or err.response.status_code != 400:
            raise
        # 400 = already credited by an earlier attempt: the coins are there, carry on.
    db.claims.mark_paid(claim_id) # your own bookkeeping
    db.ledger.insert(claim_id=claim_id, currency="GITA_COIN", amount=amount, external=True)
```

Claims, held coins and moving them on sign-in

The rules from [Gita coins → players who haven't signed in](#): a claim requires a signed-in player, a paid purchase never fails, and held coins move to Baziness when the player signs in.

TypeScript (Node)

```
// grants.ts: Gita coins go to the platform; COIN/GEM stay local.
export async function grantCurrency(tx: Tx, change: WalletChange & { idempotencyKey: string
},
    opts: { requireLink: boolean; metaData?: object }) {
```

```

    if (change.currency !== "GITA_COIN" || !platformEnabled() || change.amount < 0) {
        return applyWalletChange(tx, change); // local wallet
    }
    const { baziness_user_id: bazinessUserId } = await tx.one("select baziness_user_id from
users where id = $1", [change.userId]);
    if (!bazinessUserId) {
        if (opts.requireLink) throw new ApiError("PLATFORM_NOT_LINKED", "برای دریافت گیتا کوین
وارد حساب بازیس شوید");
        return applyWalletChange(tx, { ...change, note: "held until a Baziness account is linked"
}); // escrow
    }
    try {
        await addCoin(bazinessUserId, change.amount, change.idempotencyKey, { source:
change.type, ...opts.metaData });
    } catch (err) {
        throw new ApiError("PLATFORM_UNAVAILABLE", "ارتباط با بازیس برقرار نشد، دوباره تلاش کنید");
    }
    await recordExternalCredit(tx, change); // ledger row, balance
    unchanged locally
}

/** Called right after a successful link (POST /api/platform/login or /token). */
export async function flushGitaEscrow(userId: number) {
    await withTx(async (tx) => {
        const user = await tx.one("select baziness_user_id from users where id = $1", [userId]);
        const wallet = await tx.oneOrNull(
            "select balance from user_wallets where user_id = $1 and currency = 'GITA_COIN' for
update", [userId]);
        if (!user.baziness_user_id || !wallet || wallet.balance <= 0) return;
        const last = await tx.one("select max(id) as id from wallet_transactions where user_id =
$1 and currency = 'GITA_COIN'", [userId]);
        const key = `escrow:${userId}:${last.id ?? 0}`;
        await addCoin(user.baziness_user_id, wallet.balance, key, { source: "escrow" });
        await applyWalletChange(tx, { userId, currency: "GITA_COIN", amount: -wallet.balance,
            type: "REWARD_SELECTION", idempotencyKey: key, note: "moved to the Baziness account"
        });
    });
}

```

C# (.NET)

```

// Inside a service with: BazinessPlatformClient platform, PlatformSettings settings,
IUserStore users, IWalletStore wallets.
public async Task GrantAsync(DbTransaction tx, WalletChange change, bool requireLink)
{
    if (change.Currency != "GITA_COIN" || !settings.IsOn || change.Amount < 0)
    { await wallets.ApplyAsync(tx, change); return; } // local wallet

    long? bazinessUserId = await users.GetBazinessUserIdAsync(tx, change.UserId);
    if (bazinessUserId is null)
    {
        if (requireLink) throw new ApiException("PLATFORM_NOT_LINKED", "برای دریافت گیتا کوین
وارد حساب بازیس شوید");
        await wallets.ApplyAsync(tx, change with { Note = "held until a Baziness account is
linked" }); // escrow
        return;
    }
    await platform.AddCoinAsync(bazinessUserId.Value, change.Amount, change.IdempotencyKey,
new { source = change.Type });
}

```

```

        await wallets.RecordExternalCreditAsync(tx, change);
    }

    public async Task FlushEscrowAsync(long userId)
    {
        await using var tx = await db.BeginTransactionAsync();
        var bazinessUserId = await users.GetBazinessUserIdAsync(tx, userId);
        var balance = await wallets.LockBalanceAsync(tx, userId, "GITA_COIN"); // SELECT ... FOR
        UPDATE
        if (bazinessUserId is null || balance <= 0) return;
        var key = $"escrow:{userId}:{await wallets.LastTransactionIdAsync(tx, userId,
"GITA_COIN")}";
        await platform.AddCoinAsync(bazinessUserId.Value, balance, key, new { source = "escrow"
});
        await wallets.ApplyAsync(tx, new WalletChange(userId, "GITA_COIN", -balance,
"REWARD_SELECTION", key));
        await tx.CommitAsync();
    }

```

Python

```

def grant_currency(tx, change: dict, require_link: bool, meta: dict | None = None) -> None:
    if change["currency"] != "GITA_COIN" or not platform_enabled() or change["amount"] < 0:
        return apply_wallet_change(tx, change) # local wallet
    baziness_user_id = tx.scalar("select baziness_user_id from users where id = %s",
[change["userId"]])
    if not baziness_user_id:
        if require_link:
            raise ApiError("PLATFORM_NOT_LINKED", "برای دریافت گیتا کوین وارد حساب بازینس
شود")
        return apply_wallet_change(tx, {**change, "note": "held until a Baziness account is
linked"}) # escrow
    platform.add_coin(baziness_user_id, change["amount"], change["idempotencyKey"],
{"source": change["type"], **(meta or {})})
    record_external_credit(tx, change)

def flush_gita_escrow(user_id: int) -> None:
    with db.transaction() as tx:
        baziness_user_id = tx.scalar("select baziness_user_id from users where id = %s",
[user_id])
        balance = tx.scalar("select balance from user_wallets where user_id = %s and currency
= 'GITA_COIN' "
                             "for update", [user_id]) or 0
        if not baziness_user_id or balance <= 0:
            return
        last = tx.scalar("select max(id) from wallet_transactions where user_id = %s and
currency = 'GITA_COIN'",
                         [user_id]) or 0
        key = f"escrow:{user_id}:{last}"
        platform.add_coin(baziness_user_id, balance, key, {"source": "escrow"})
        apply_wallet_change(tx, {"userId": user_id, "currency": "GITA_COIN", "amount": -
balance,
                                "type": "REWARD_SELECTION", "idempotencyKey": key})

```

Baziness Pay client

Steps 1–5 of Baziness Pay: the payment page URL, `get_payment`, `verify_payment` (a **form** body) and `consume`.

TypeScript (Node)

```
// platform/baziness-pay.ts
const BASE = (process.env.BAZINESS_PAY_BASE_URL ??
"https://pay.baziness.com").replace(/\/+$/, "");
const PNAME = process.env.BAZINESS_PAY_PACKAGE_NAME ?? "";
const PREFIX = process.env.BAZINESS_SKU_PREFIX ?? "";
export const payEnabled = () => process.env.BAZINESS_PAY_ENABLED === "true";

export const skuFor = (productSku: string, override?: string | null) => override ||
`${PREFIX}${productSku}`;

export const buildBuyUrl = (uid: string, sku: string) =>
`${BASE}/buy?${new URLSearchParams({ sku, uid, pname: PNAME })}`;

/** The verification token of a completed, unconsumed payment – or null while unpaid. */
export async function getPayment(uid: string, sku: string): Promise<{ token: string; sku:
string } | null> {
  try {
    const res = await fetch(`${BASE}/get_payment`, {
      method: "POST", headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ uid, sku, pname: PNAME }),
    });
    if (!res.ok) return null;
    const j = (await res.json()) as { success?: boolean; token?: string; sku?: string };
    return j.success && j.token ? { token: j.token, sku: j.sku ?? sku } : null;
  } catch { return null; }
}

/** The authoritative check. Never trust a token from the client. */
export async function verifyPayment(uid: string, token: string, sku: string):
Promise<boolean> {
  try {
    const res = await fetch(`${BASE}/verify_payment`, {
      method: "POST", headers: { "Content-Type": "application/x-www-form-urlencoded" }, //
form, not JSON
      body: new URLSearchParams({ uid, token, sku, pname: PNAME }).toString(),
    });
    if (!res.ok) return false;
    const j = (await res.json()) as { success?: boolean; error?: string };
    if (!j.success && j.error) console.warn("[pay] verify_payment rejected:", j.error);
    return j.success === true;
  } catch { return false; }
}

/** Let the SKU be bought again. Best effort, 3 attempts; never blocks an already-granted
purchase. */
export async function consumePayment(uid: string, sku: string): Promise<boolean> {
  for (let attempt = 1; attempt <= 3; attempt++) {
    try {
      const res = await fetch(`${BASE}/consume`, {
        method: "POST", headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ uid, sku, pname: PNAME }),
      });
    }
  }
}
```

```

        if (res.ok && ((await res.json()) as { success?: boolean }).success) return true;
    } catch { /* retry */ }
}
console.error("[pay] consume failed uid=%s sku=%s", uid, sku);
return false;
}

```

C# (.NET)

```

// BazinessPay.cs - register with builder.Services.AddHttpClient<BazinessPay>();
using System.Net.Http.Json;

public sealed class BazinessPay(HttpClient http, IConfiguration cfg, ILogger<BazinessPay>
log)
{
    string Base => (cfg["BazinessPay:BaseUrl"] ?? "https://pay.baziness.com").TrimEnd('/');
    string PName => cfg["BazinessPay:PackageName"]!;
    public bool Enabled => cfg.GetValue<bool>("BazinessPay:Enabled");
    public string SkuFor(string productSku, string? overrideSku = null)
        => string.IsNullOrEmpty(overrideSku) ? $"{cfg["BazinessPay:SkuPrefix"]}{productSku}"
: overrideSku;

    public string BuildBuyUrl(string uid, string sku) =>
        $"{Base}/buy?sku={Uri.EscapeDataString(sku)}&uid={Uri.EscapeDataString(uid)}&pname=
{Uri.EscapeDataString(PName)}";

    record PaymentDto(bool Success, string? Token, string? Sku);
    record ResultDto(bool Success, string? Error);

    public async Task<(string Token, string Sku)?> GetPaymentAsync(string uid, string sku)
    {
        try
        {
            var res = await http.PostAsJsonAsync($"{Base}/get_payment", new { uid, sku, pname
= PName });
            if (!res.IsSuccessStatusCode) return null;
            var j = await res.Content.ReadFromJsonAsync<PaymentDto>();
            return j is { Success: true, Token: { Length: > 0 } t } ? (t, j.Sku ?? sku) :
null;
        }
        catch (Exception ex) { log.LogError(ex, "get_payment failed"); return null; }
    }

    public async Task<bool> VerifyPaymentAsync(string uid, string token, string sku)
    {
        try
        {
            var form = new FormUrlEncodedContent(new Dictionary<string, string> // form,
not JSON
            { ["uid"] = uid, ["token"] = token, ["sku"] = sku, ["pname"] = PName });
            var res = await http.PostAsync($"{Base}/verify_payment", form);
            if (!res.IsSuccessStatusCode) return false;
            var j = await res.Content.ReadFromJsonAsync<ResultDto>();
            if (j is { Success: false, Error: not null }) log.LogWarning("verify_payment
rejected: {Error}", j.Error);
            return j?.Success == true;
        }
        catch (Exception ex) { log.LogError(ex, "verify_payment failed"); return false; }
    }
}

```

```

public async Task<bool> ConsumeAsync(string uid, string sku)
{
    for (var attempt = 1; attempt <= 3; attempt++)
    {
        try
        {
            var res = await http.PostAsJsonAsync($"{Base}/consume", new { uid, sku, pname
= PName });
            if (res.IsSuccessStatusCode && (await
res.Content.ReadFromJsonAsync<ResultDto>())?.Success == true) return true;
        }
        catch (Exception ex) { log.LogWarning(ex, "consume attempt {Attempt} failed",
attempt); }
    }
    log.LogError("consume failed uid={Uid} sku={Sku}", uid, sku);
    return false;
}
}

```

Python

```

# baziness/pay.py
import os
import requests
from urllib.parse import urlencode

BASE = os.getenv("BAZINESS_PAY_BASE_URL", "https://pay.baziness.com").rstrip("/")
PNAME = os.getenv("BAZINESS_PAY_PACKAGE_NAME", "")
PREFIX = os.getenv("BAZINESS_SKU_PREFIX", "")

def pay_enabled() -> bool:
    return os.getenv("BAZINESS_PAY_ENABLED", "false").lower() == "true"

def sku_for(product_sku: str, override: str | None = None) -> str:
    return override or f"{PREFIX}{product_sku}"

def build_buy_url(uid: str, sku: str) -> str:
    return f"{BASE}/buy?{urlencode({'sku': sku, 'uid': uid, 'pname': PNAME})}"

def get_payment(uid: str, sku: str) -> dict | None:
    try:
        r = requests.post(f"{BASE}/get_payment", json={"uid": uid, "sku": sku, "pname":
PNAME}, timeout=10)
        j = r.json() if r.ok else {}
        return {"token": j["token"], "sku": j.get("sku", sku)} if j.get("success") and
j.get("token") else None
    except requests.RequestException:
        return None

def verify_payment(uid: str, token: str, sku: str) -> bool:
    try: # form-encoded, not JSON
        r = requests.post(f"{BASE}/verify_payment", data={"uid": uid, "token": token, "sku":
sku, "pname": PNAME},

```

```

        timeout=10)
    return r.ok and r.json().get("success") is True
except requests.RequestException:
    return False

def consume_payment(uid: str, sku: str) -> bool:
    for _ in range(3):
        try:
            r = requests.post(f"{BASE}/consume", json={"uid": uid, "sku": sku, "pname":
PNAME}, timeout=10)
            if r.ok and r.json().get("success"):
                return True
        except requests.RequestException:
            pass
    return False

```

The payments table

Schema (PostgreSQL)

```

create table payments (
    id                bigserial primary key,
    user_id           bigint not null references users(id),
    shop_product_id   bigint not null references shop_products(id),
    provider           text not null default 'BAZINESS',
    provider_sku       text not null,
    -- The Baziness payment token; unique so the same payment can never be fulfilled twice.
    provider_transaction_id text,
    amount            bigint not null,
    currency_code      text not null,
    status             text not null default 'CREATED' check (status in
('CREATED', 'REDIRECTED', 'PAID', 'VERIFIED', 'FAILED', 'CANCELLED', 'FULFILLED')),
    reward_snapshot    jsonb not null,          -- what the player was promised when they
clicked
    failure_reason     text,
    created_at         timestamptz not null default now(),
    redirected_at      timestamptz,
    verified_at        timestamptz,
    fulfilled_at       timestamptz,
    updated_at         timestamptz not null default now(),
    unique (provider, provider_transaction_id)
);
create index payments_user on payments(user_id, created_at desc);

```

Purchase endpoints

`POST /api/shop/purchase/start` and `POST /api/shop/purchase/verify`: save the order, then verify and give the item exactly once.

TypeScript (Node)

```

// services/shop.ts
export async function startPurchase(userId: number, productId: number) {
  if (!payEnabled()) throw new ApiError("PAYMENT_DISABLED", "پرداخت به‌زودی فعال می‌شود");
  const product = await db.one("select * from shop_products where id = $1 and enabled",
    [productId]);
  const sku = skuFor(product.sku, product.baziness_product_id);
  const payment = await db.one(
    `insert into payments (user_id, shop_product_id, provider_sku, amount, currency_code,
    status, reward_snapshot, redirected_at)
    values ($1, $2, $3, $4, $5, 'REDIRECTED', $6, now()) returning id`,
    [userId, product.id, sku, product.price, product.currency_code,
    JSON.stringify(product.grants(product))]);
  return { paymentId: String(payment.id), url: buildBuyUrl(String(userId), sku) };
}

export async function verifyPurchase(userId: number, paymentId: number) {
  const p = await db.oneOrNull("select * from payments where id = $1", [paymentId]);
  if (!p || p.user_id !== userId) throw notFound("payment");
  if (p.status === "FULFILLED" || p.status === "FAILED" || p.status === "CANCELLED") {
    return { status: p.status, wallets: await getBalances(userId), granted: [] };
  }
  const uid = String(userId);
  const found = await getPayment(uid, p.provider_sku);
  if (!found) return { status: p.status, wallets: await getBalances(userId), granted: [] };
  // not paid yet
  if (!(await verifyPayment(uid, found.token, p.provider_sku))) {
    await db.none("update payments set failure_reason = 'verify_payment rejected' where id =
    $1 and status <> 'FULFILLED'", [paymentId]);
    throw new ApiError("PAYMENT_FAILED", "پرداخت تأیید نشد");
  }
  const granted = await fulfilPayment(paymentId, found.token);
  void consumePayment(uid, p.provider_sku); // after commit, best effort
  return { status: "FULFILLED", wallets: await getBalances(userId), granted };
}

export function fulfilPayment(paymentId: number, providerToken: string) {
  return withTx(async (tx) => {
    const p = await tx.one("select * from payments where id = $1 for update", [paymentId]);
    if (p.status === "FULFILLED") return [];
    const dup = await tx.oneOrNull(
      "select id from payments where provider = 'BAZINESS' and provider_transaction_id = $1
      and id <> $2", [providerToken, paymentId]);
    if (dup) {
      await tx.none("update payments set status = 'FAILED', failure_reason = 'duplicate
      provider token' where id = $1", [paymentId]);
      return [];
    }
    await tx.none("update payments set status = 'VERIFIED', provider_transaction_id = $2,
    verified_at = now() where id = $1",
    [paymentId, providerToken]);
    for (const g of p.reward_snapshot) {
      await grantCurrency(tx, { user_id: p.user_id, currency: g.currency, amount: g.amount,
      type: "SHOP_PURCHASE",
      idempotencyKey: `payment:${paymentId}:${g.currency}` }, { requireLink: false }); //
      Gita coins: escrow if not linked
    }
    await tx.none("update payments set status = 'FULFILLED', fulfilled_at = now() where id =
    $1", [paymentId]);
  });
}

```

```

    return p.reward_snapshot;
});
}

```

C# (.NET)

```

app.MapPost("/api/shop/purchase/start", async (StartRequest body, ClaimsPrincipal me,
BazinessPay pay, ShopDb db) =>
{
    if (!pay.Enabled) return Results.Ok(new { error = "PAYMENT_DISABLED" });
    var userId = me.UserId();
    var product = await db.GetEnabledProductAsync(body.ProductId);
    if (product is null) return Results.NotFound();
    var sku = pay.SkuFor(product.Sku, product.BazinessProductId);
    var paymentId = await db.InsertPaymentAsync(userId, product, sku, status: "REDIRECTED");
    return Results.Ok(new { paymentId = paymentId.ToString(), url =
pay.BuildBuyUrl(userId.ToString(), sku) });
});

app.MapPost("/api/shop/purchase/verify", async (VerifyRequest body, ClaimsPrincipal me,
BazinessPay pay, ShopDb db) =>
{
    var userId = me.UserId();
    var p = await db.GetPaymentAsync(body.PaymentId);
    if (p is null || p.UserId != userId) return Results.NotFound();
    if (p.Status is "FULFILLED" or "FAILED" or "CANCELLED")
        return Results.Ok(new { status = p.Status, wallets = await db.BalancesAsync(userId),
granted = Array.Empty<object>() });

    var uid = userId.ToString();
    var found = await pay.GetPaymentAsync(uid, p.ProviderSku);
    if (found is null) // not paid yet – keep
waiting
        return Results.Ok(new { status = p.Status, wallets = await db.BalancesAsync(userId),
granted = Array.Empty<object>() });
    if (!await pay.VerifyPaymentAsync(uid, found.Value.Token, p.ProviderSku))
    {
        await db.SetFailureReasonAsync(p.Id, "verify_payment_rejected");
        return Results.Ok(new { error = "PAYMENT_FAILED" });
    }

    var granted = await db.FulfilAsync(p.Id, found.Value.Token); // one tx: FOR UPDATE,
dup-token check, grants, FULFILLED
    _ = pay.ConsumeAsync(uid, p.ProviderSku); // after commit, best
effort
    return Results.Ok(new { status = "FULFILLED", wallets = await db.BalancesAsync(userId),
granted });
});

public record StartRequest(long ProductId);
public record VerifyRequest(long PaymentId);

```

Python

```

from fastapi import Depends, HTTPException
from baziness import pay

@app.post("/api/shop/purchase/start")

```

```

def start_purchase(body: dict, user=Depends(current_user)):
    if not pay.pay_enabled():
        raise HTTPException(400, "PAYMENT_DISABLED")
    product = db.fetch_one("select * from shop_products where id = %s and enabled",
[body["productId"]])
    if not product:
        raise HTTPException(404)
    sku = pay.sku_for(product["sku"], product["baziiness_product_id"])
    payment_id = db.scalar(
        """insert into payments (user_id, shop_product_id, provider_sku, amount,
currency_code, status,
                                reward_snapshot, redirected_at)
        values (%s, %s, %s, %s, %s, 'REDIRECTED', %s, now()) returning id""",
        [user.id, product["id"], sku, product["price"], product["currency_code"],
json.dumps(product_grants(product))])
    return {"paymentId": str(payment_id), "url": pay.build_buy_url(str(user.id), sku)}

@app.post("/api/shop/purchase/verify")
def verify_purchase(body: dict, user=Depends(current_user)):
    p = db.fetch_one("select * from payments where id = %s", [body["paymentId"]])
    if not p or p["user_id"] != user.id:
        raise HTTPException(404)
    if p["status"] in ("FULFILLED", "FAILED", "CANCELLED"):
        return {"status": p["status"], "wallets": balances(user.id), "granted": []}
    uid = str(user.id)
    found = pay.get_payment(uid, p["provider_sku"])
    if not found: # not paid yet
        return {"status": p["status"], "wallets": balances(user.id), "granted": []}
    if not pay.verify_payment(uid, found["token"], p["provider_sku"]):
        db.execute("update payments set failure_reason = 'verify_payment rejected' where id =
%s "
                    "and status <> 'FULFILLED'", [p["id"]])
        raise HTTPException(400, "PAYMENT_FAILED")
    granted = fulfil_payment(p["id"], found["token"]) # one tx: FOR UPDATE, dup-token
check, grants, FULFILLED
    threading.Thread(target=pay.consume_payment, args=(uid, p["provider_sku"]),
daemon=True).start()
    return {"status": "FULFILLED", "wallets": balances(user.id), "granted": granted}

```

Local fake gateway

The example game includes a stand-in for Baziness Pay that speaks the same protocol, for working without gateway access. It is switched on with these settings, and never in production:

```
.env (local only)
```

```

DEV_FAKE_BAZINESS_PAY=true
BAZINESS_PAY_ENABLED=true
BAZINESS_PAY_BASE_URL=http://localhost:3390/dev/baziness

```

First-touch attribution

The rules from [Links & UTM](#). The first tab is the **client** side, reading the tags and removing them from the address. The others save them on the **server** when the player is created.

TypeScript (client)

```
// api/client.ts - the FIRST module the app imports.
const UTM_KEY = "mygame_utm";
const TAG_KEY = "mygame_tag";
const UTM_FIELDS = ["source", "medium", "campaign", "content"] as const;
type Utm = Partial<Record<(typeof UTM_FIELDS)[number], string>>;

const clean = (raw: unknown): string | null => {
  if (typeof raw !== "string") return null;
  const v = raw.trim().toLowerCase().replace(/\\s+/g, "_").slice(0, 64);
  return /^[a-z0-9][a-z0-9._-]*$/i.test(v) ? v : null; // ASCII only
};

export const FIRST_TOUCH_UTM: Utm = (() => {
  let stored: Utm = {};
  try { stored = JSON.parse(localStorage.getItem(UTM_KEY) ?? "{}"); } catch {}
  try {
    const query = new URLSearchParams(location.search);
    const keys = [...query.keys()].filter((k) => k.toLowerCase().startsWith("utm_"));
    const fromUrl: Utm = {};
    for (const k of keys) {
      const field = k.toLowerCase().slice(4) as keyof Utm;
      const v = clean(query.get(k));
      if (UTM_FIELDS.includes(field as never) && v) fromUrl[field] = v;
    }
    if (keys.length) { // take them out of the address
      keys.forEach((k) => query.delete(k));
      const search = query.toString();
      history.replaceState(history.state, "", `${location.pathname}${search ? `?${search}` : ""}${location.hash}`);
    }
    if (Object.keys(fromUrl).length && !Object.keys(stored).length) { // first touch wins
      stored = fromUrl;
      localStorage.setItem(UTM_KEY, JSON.stringify(stored));
    }
  } catch { /* storage blocked: the visit is simply untagged */ }
  return stored;
})();

export function firstTouchTag(): string | null {
  try {
    const fromUrl = clean(new URLSearchParams(location.search).get("tag"));
    if (fromUrl && !localStorage.getItem(TAG_KEY)) localStorage.setItem(TAG_KEY, fromUrl);
    return localStorage.getItem(TAG_KEY) ?? fromUrl;
  } catch { return null; }
}

// Sent with the device login AND with /api/platform/token:
// { deviceId, tag: firstTouchTag(), utm: FIRST_TOUCH_UTM }
```

C# (server)

```

using System.Text.RegularExpressions;

// Save attribution ONLY when the account is created (first touch). Four separate columns.
public async Task<User> EnsureUserByDeviceAsync(string deviceId, string? tag, UtmParams? utm)
{
    const string sql = """
        insert into users (device_id, acq_tag, utm_source, utm_medium, utm_campaign,
utm_content)
        values (@deviceId, @tag, @source, @medium, @campaign, @content)
        on conflict (device_id) do update set last_seen = now() -- existing account:
attribution untouched
        returning *;
        """;
    return await db.QuerySingleAsync<User>(sql, new {
        deviceId, tag = Utm.Clean(tag),
        source = Utm.Clean(utm?.Source), medium = Utm.Clean(utm?.Medium),
        campaign = Utm.Clean(utm?.Campaign), content = Utm.Clean(utm?.Content),
    });
}

public static class Utm
{
    static readonly Regex Valid = new("^[a-z0-9][a-z0-9._-]*$", RegexOptions.Compiled);
    public static string? Clean(string? raw)
    {
        if (raw is null) return null;
        var v = Regex.Replace(raw.Trim().ToLowerInvariant(), @"\s+", "_");
        if (v.Length > 64) v = v[..64];
        return Valid.IsMatch(v) ? v : null;
    }
}

public record UtmParams(string? Source, string? Medium, string? Campaign, string? Content);

```

Python (server)

```

import re

_VALID = re.compile(r"^[a-z0-9][a-z0-9._-]*$")

def clean_utm(raw: str | None) -> str | None:
    if not isinstance(raw, str):
        return None
    v = re.sub(r"\s+", "_", raw.strip().lower())[:64]
    return v if _VALID.match(v) else None

def ensure_user_by_device(device_id: str, tag: str | None, utm: dict | None):
    """Attribution is saved ONLY when the account is created (first touch)."""
    utm = utm or {}
    return db.fetch_one(
        """
        insert into users (device_id, acq_tag, utm_source, utm_medium, utm_campaign,
utm_content)
        values (%s, %s, %s, %s, %s, %s)
        on conflict (device_id) do update set last_seen = now()
        returning *
        """,

```

```
[device_id, clean_utm(tag), *(clean_utm(utm.get(k) or utm.get(f"utm_{k}"))
    for k in ("source", "medium", "campaign", "content"))],
)
```

Schema (PostgreSQL)

```
alter table users
  add column if not exists acq_tag      text,
  add column if not exists utm_source   text,
  add column if not exists utm_medium  text,
  add column if not exists utm_campaign text,
  add column if not exists utm_content text;

create index if not exists idx_users_utm_combo on users (utm_source, utm_medium,
utm_campaign, utm_content)
  where utm_source is not null or utm_medium is not null or utm_campaign is not null or
utm_content is not null;
```

Building links

Baziness page links ([parameters](#)):

TypeScript

```
type PageOptions = { redirect?: "shop" | "profile"; allowGuest?: boolean; gameId?: string;
buildStore?: string };

export function bazinessPageUrl(webBase: string, path: string, packageName: string, token:
string | null, o: PageOptions = {}) {
  const u = new URL(webBase.replace(/\/+$/, "") + (path || "/"));
  if (o.gameId) u.searchParams.set("game_id", o.gameId);
  u.searchParams.set("packageName", packageName);
  u.searchParams.set("sdkType", "js"); // never optional
  if (token) u.searchParams.set("linkedToken", token);
  if (o.redirect) u.searchParams.set("redirect", o.redirect);
  if (o.allowGuest === false) u.searchParams.set("allowGuest", "false");
  if (o.buildStore) u.searchParams.set("buildStore", o.buildStore);
  return u.toString();
}

// bazinessPageUrl(webBase, "/shop", pkg, token, { gameId: "example", redirect: "shop" })
// bazinessPageUrl(webBase, "/", pkg, token, { allowGuest: false })
```

C#

```
using System.Web; // or Microsoft.AspNetCore.WebUtilities.QueryHelpers

public static string BazinessPageUrl(string webBase, string path, string packageName, string?
token,
  string? redirect = null, bool allowGuest = true, string? gameId = null, string?
buildStore = null)
{
  var q = HttpUtility.ParseQueryString(string.Empty); // encodes values for you
  if (gameId != null) q["game_id"] = gameId;
```

```

q["packageName"] = packageName;
q["sdkType"] = "js"; // never optional
if (!string.IsNullOrEmpty(token)) q["linkedToken"] = token;
if (redirect != null) q["redirect"] = redirect;
if (!allowGuest) q["allowGuest"] = "false";
if (buildStore != null) q["buildStore"] = buildStore;
return $"{webBase.TrimEnd('/')}{(string.IsNullOrEmpty(path) ? "/" : path)}{q}";
}

// BazinessPageUrl(webBase, "/shop", pkg, token, redirect: "shop", gameName: "example")

```

Python

```

from urllib.parse import urlencode

def baziness_page_url(web_base: str, path: str, package_name: str, token: str | None, *,
                      redirect: str | None = None, allow_guest: bool = True,
                      game_name: str | None = None, build_store: str | None = None) -> str:
    params: dict[str, str] = {}
    if game_name:
        params["game_name"] = game_name
    params["packageName"] = package_name
    params["sdkType"] = "js" # never optional
    if token:
        params["linkedToken"] = token
    if redirect:
        params["redirect"] = redirect
    if not allow_guest:
        params["allowGuest"] = "false"
    if build_store:
        params["buildStore"] = build_store
    return f"{web_base.rstrip('/')}{path or '/'}{urlencode(params)}"

# baziness_page_url(web_base, "/shop", pkg, token, game_name="example", redirect="shop")

```

Campaign links for marketing:

TypeScript

```

export function campaignLink(gameUrl: string, utm: { source: string; medium: string;
campaign: string; content?: string }) {
    const u = new URL(gameUrl);
    u.searchParams.set("utm_source", utm.source);
    u.searchParams.set("utm_medium", utm.medium);
    u.searchParams.set("utm_campaign", utm.campaign);
    if (utm.content) u.searchParams.set("utm_content", utm.content);
    return u.toString();
}

// campaignLink("https://yourgame.example.com/", { source: "baziness", medium: "inapp",
campaign: "shahrivar_1405", content: "banner_home" })

```

C#

```

public static string CampaignLink(string gameUrl, string source, string medium, string
campaign, string? content = null)
{

```

```
var q = System.Web.HttpUtility.ParseQueryString(string.Empty);
q["utm_source"] = source; q["utm_medium"] = medium; q["utm_campaign"] = campaign;
if (content != null) q["utm_content"] = content;
return $"{gameUrl.TrimEnd('/')}/?{q}";
}
```

Python

```
from urllib.parse import urlencode

def campaign_link(game_url: str, source: str, medium: str, campaign: str, content: str | None = None) -> str:
    params = {"utm_source": source, "utm_medium": medium, "utm_campaign": campaign}
    if content:
        params["utm_content"] = content
    return f"{game_url.rstrip('/')}/?{urlencode(params)}"
```

Client code

Ready-made client code — the platform controller, page messages, the coin balance, the reward button, the tutorial offer page and the purchase window — for the web, React, Cocos Creator and Unity.

These samples are the client side of the [guides](#). Most come as **TypeScript (web)**, **React** and **C# (Unity)** tabs, and there's a complete [Cocos Creator](#) version at the end. The in-game web view (the bottom sheet) has [its own page](#).

Platform controller

The heart of the client: it asks your server for the player's token, builds every page link, opens pages, and runs the first-launch choice on the splash ([Login](#)).

TypeScript (web)

```
// baziness.ts - the client controller.
const TOKEN_KEY = "mygame_baziness_token";
const CHOICE_KEY = "mygame_platform_choice";

export class BazinessPlatform {
  enabled = false;
  isLinked = false;
  private token: string | null = null;
  private webBase = "";
  private apiBase = "";
  private packageName = "";
  private loginGate: ((ok: boolean) => void) | null = null;
  private loginSucceeded = false;
  onChange: (() => void) | null = null;

  constructor(private sheet: BazinessSheet, private deviceId: string) {
    sheet.onMessage = (m) => this.handleMessage(m);
    sheet.onClosed = () => this.handleClosed();
  }

  get choiceMade() { return localStorage.getItem(CHOICE_KEY) === "1"; }
  markChoiceMade() { localStorage.setItem(CHOICE_KEY, "1"); }

  async loadConfig(): Promise<boolean> {
    try {
      const cfg = await fetch("/api/platform/config").then((r) => r.json());
      this.enabled = cfg.enabled;
      if (cfg.enabled) this.apply(cfg);
    } catch { this.enabled = false; }
    return this.enabled;
  }

  /** Resolve the token. Returns false when the page is about to reload into another account.
```

```

*/
    async init(): Promise<boolean> {
        // The Baziness app opens games with ?linkedToken=... - forward it once, then it is
        stripped on reload.
        const linkedToken = new URLSearchParams(location.search).get("linkedToken") ||
        localStorage.getItem(TOKEN_KEY);
        const resp = await fetch("/api/platform/token", {
            method: "POST", headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ deviceId: this.deviceId, linkedToken }),
        }).then((r) => r.json()).catch(() => null);
        if (!resp?.enabled) { this.enabled = false; return true; }
        this.enabled = true;
        this.apply(resp);
        if (resp.switchToDeviceId && resp.switchToDeviceId !== this.deviceId) {
            this.switchAccount(resp.switchToDeviceId);
            return false;
        }
        if (resp.token) {
            this.token = resp.token;
            this.isLinked = !!resp.isLinkedToAccount;
            localStorage.setItem(TOKEN_KEY, resp.token);
        }
        this.onChange?.();
        return true;
    }

    /** Splash "Sign in with Baziness": resolves true once a login/link succeeded, false if
    closed. */
    async openLoginAndWait(): Promise<boolean> {
        if (!this.token) await this.init(); // the anonymous user's token becomes linkedToken
        if (!this.enabled || !this.webBase) return false;
        return new Promise((resolve) => {
            this.loginGate = resolve;
            this.loginSucceeded = false;
            this.sheet.open(this.pageUrl("/", { allowGuest: false }));
        });
    }

    /** Settings → "Connect your account": the profile page. */
    openAccount() { this.sheet.open(this.pageUrl("/profile", { redirect: "profile", allowGuest:
    false })); }

    pageUrl(path: string, o: { redirect?: string; allowGuest?: boolean; extra?: Record<string,
    string> } = {}) {
        const u = new URL(this.webBase + path);
        for (const [k, v] of Object.entries(o.extra ?? {})) u.searchParams.set(k, v);
        u.searchParams.set("packageName", this.packageName);
        u.searchParams.set("sdkType", "js");
        if (this.token) u.searchParams.set("linkedToken", this.token);
        if (o.redirect) u.searchParams.set("redirect", o.redirect);
        if (o.allowGuest === false) u.searchParams.set("allowGuest", "false");
        return u.toString();
    }

    private apply(c: { webBase?: string; apiBase?: string; packageName?: string }) {
        this.webBase = c.webBase ?? ""; this.apiBase = c.apiBase ?? ""; this.packageName =
        c.packageName ?? "";
    }

    // handleMessage / handleClosed / switchAccount: see "Messages from the Baziness page"

```

```
below.  
}
```

React

```
// Splash.tsx - the first-time choice.  
export function Splash({ onReady }: { onReady: (me: Me) => void }) {  
  const [needsChoice, setNeedsChoice] = useState(false);  
  const choice = useRef<() => void | null>(null);  
  const loggedIn = useRef(false);  
  
  useEffect(() => {  
    (async () => {  
      let me = (await authenticate()).me; // 1. your own account first  
      if (await baziness.loadConfig()) { // 2. platform on?  
        if (!baziness.choiceMade && !me.user.isBazinessLinked) {  
          setNeedsChoice(true); // 3. ask once  
          await new Promise<void>((r) => (choice.current = r));  
          setNeedsChoice(false);  
        }  
        if (!(await baziness.init())) return; // 4. reloading into the linked  
account  
        if (loggedIn.current || baziness.isLinked !== me.user.isBazinessLinked) {  
          me = await api<Me>("GET", "/api/me"); // 5. show the linked account  
        }  
      }  
      onReady(me);  
    })();  
  }, [onReady]);  
  
  const chooseGuest = () => { baziness.markChoiceMade(); choice.current?.(); };  
  const chooseLogin = async () => {  
    await baziness.init();  
    if (!(await baziness.openLoginAndWait())) return; // closed without logging in → keep  
the buttons  
    loggedIn.current = true;  
    choice.current?.();  
  };  
  
  return needsChoice ? (  
    <div className="splash-choice">  
      <button className="btn btn-green" onClick={chooseLogin}>ورود با حساب بازینس</button>  
      <button className="btn btn-ghost" onClick={chooseGuest}>ادامه به عنوان مهمان</button>  
    </div>  
  ) : <ProgressBar />;  
}
```

C# (Unity)

```
// BazinessLogin.cs - the same protocol from a Unity client. Your SERVER still does every  
// platform call; Unity only talks to your server and shows the page in a WebView.  
using System;  
using System.Collections;  
using UnityEngine;  
using UnityEngine.Networking;  
  
public class BazinessLogin : MonoBehaviour  
{  
  const string TokenKey = "baziness_token", ChoiceKey = "baziness_choice";
```

```

[SerializeField] string serverUrl = "https://api.yourgame.com";
[SerializeField] BazinessSheet sheet; // see the Bottom sheet page

public bool Enabled { get; private set; }
public bool IsLinked { get; private set; }
public string Token { get; private set; }
public string PackageName => packageName;
public string GameOfferUrl { get; private set; } // from /api/platform/token
string webBase, apiBase, packageName;
Action<bool> loginGate;

[Serializable] class TokenResponse {
    public bool enabled; public string webBase, apiBase, packageName, token,
switchToDeviceId;
    public long bazinessUserId; public bool isLinkedToAccount;
}

public IEnumerator Init(string deviceId)
{
    var body = $"{{"deviceId":{"deviceId"},"linkedToken":
Quote(PlayerPrefs.GetString(TokenKey, null))}}";
    using var req = new UnityWebRequest($"{serverUrl}/api/platform/token", "POST");
    req.uploadHandler = new UploadHandlerRaw(System.Text.Encoding.UTF8.GetBytes(body));
    req.downloadHandler = new DownloadHandlerBuffer();
    req.SetRequestHeader("Content-Type", "application/json");
    yield return req.SendWebRequest();
    if (req.result != UnityWebRequest.Result.Success) { Enabled = false; yield break; }

    var r = JsonUtility.FromJson<TokenResponse>(req.downloadHandler.text);
    Enabled = r.enabled;
    if (!r.enabled) yield break;
    (webBase, apiBase, packageName) = (r.webBase, r.apiBase, r.packageName);
    if (!string.IsNullOrEmpty(r.switchToDeviceId) && r.switchToDeviceId != deviceId)
    {
        DeviceIdentity.Set(r.switchToDeviceId); // adopt the other account ...
        PlayerPrefs.DeleteKey(TokenKey);
        GameBoot.Restart(); // ... and boot again as that player
        yield break;
    }
    if (!string.IsNullOrEmpty(r.token)) { Token = r.token; IsLinked =
r.isLinkedToAccount; PlayerPrefs.SetString(TokenKey, r.token); }
}

public void OpenLogin(Action<bool> done)
{
    loginGate = done;
    sheet.Open(PageUrl("/", allowGuest: false), OnRawPageMessage, onClosed: () => {
loginGate?.Invoke(IsLinked); loginGate = null; });
}

public string PageUrl(string path, string redirect = null, bool allowGuest = true, string
extra = null)
{
    var url = $"{webBase}{path}?{extra}{(extra != null ? "&" : "")}packageName=
Uri.EscapeDataString(packageName)&sdkType=js";
    if (!string.IsNullOrEmpty(Token)) url += "&linkedToken=
Uri.EscapeDataString(Token)";
    if (redirect != null) url += "&redirect={redirect}";
    if (!allowGuest) url += "&allowGuest=false";
    return url;
}

```

```

    }

    public void OnRawPageMessage(string json) { /* see "Messages from the Baziness page"
below */ }
    static string Quote(string s) => string.IsNullOrEmpty(s) ? "null" : $"\"{s}\"";
}

```

Handling page messages

`close`, `login`, `token` and `logout`, as listed in [Baziness pages → messages](#).

TypeScript (web)

```

// Inside BazinessPlatform (continued).
const extractToken = (data: unknown): string | undefined =>
    typeof data === "string" ? data
    : data && typeof (data as { token?: unknown }).token === "string" ? (data as { token:
string }).token
    : undefined;

handleMessage(msg: { type: string; data?: unknown }) {
    const token = extractToken(msg.data);
    switch (msg.type) {
        case "close":
            this.sheet.close();
            break;
        case "login":
            if (token) void this.applyNewLogin(token); // signed into an account → switch or link
            else void this.linkedInPlace(); // linked onto the current player
            break;
        case "token":
            break; // silent handshake – NOT a login
        case "logout":
            this.sheet.close();
            localStorage.clear(); // identity, token, choice, tutorial
            flags...
            this.reloadClean(); // next boot = brand-new visitor
            break;
    }
}

private async applyNewLogin(token: string) {
    const result = await fetch("/api/platform/login", {
        method: "POST", headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId, token }),
    }).then((r) => r.json());
    if (result.switchToDeviceId && result.switchToDeviceId !== this.deviceId) {
        this.markChoiceMade();
        this.switchAccount(result.switchToDeviceId); // reloads
        return;
    }
    if (result.ok) {
        localStorage.setItem(TOKEN_KEY, token);
        await this.init(); // fresh token + linked flag
        this.loginSucceeded = true;
        this.markChoiceMade();
        this.sheet.close(); // handleClosed resolves the login gate
    }
}

```

```

    }
    // on failure keep the page open so the player can retry
  }

  private async linkedInPlace() {
    this.isLinked = true;
    this.loginSucceeded = true;
    this.markChoiceMade();
    if (this.loginGate) { await this.init(); this.sheet.close(); }
    this.onChange?.();
  }

  private handleClosed() {
    const gate = this.loginGate;
    this.loginGate = null;
    gate?.(this.loginSucceeded); // true only if a login actually happened
    this.onChange?.(); // coin balance / link state may have
    changed
  }

  private switchAccount(deviceId: string) {
    setDeviceId(deviceId); // your own identity store
    localStorage.removeItem(TOKEN_KEY);
    this.reloadClean();
  }

  private reloadClean() {
    const url = new URL(location.href);
    url.searchParams.delete("linkedToken"); // never re-apply a stale token
    url.searchParams.delete("device"); // a ?device= test override would undo the
    switch
    location.replace(url.toString());
  }

```

React

```

// BazinessSheet.tsx – listening for page messages (only from the page's own origin).
function parse(data: unknown): { type: string; data?: unknown } | null {
  let value = data;
  if (typeof data === "string") {
    try { value = JSON.parse(data); } catch { return null; }
  }
  return value && typeof value === "object" && typeof (value as { type?: unknown }).type ===
    "string"
    ? (value as { type: string; data?: unknown })
    : null;
}

useEffect(() => {
  if (!url) return;
  const origin = new URL(url).origin;
  const onMessage = (event: MessageEvent) => {
    if (event.origin !== origin) return; // ignore HMR, extensions, other frames
    const msg = parse(event.data);
    if (msg) baziness.handleMessage(msg);
  };
  window.addEventListener("message", onMessage);

```

```
return () => window.removeEventListener("message", onMessage);
}, [url]);
```

C# (Unity)

```
// Called by BazinessSheet with the raw JSON the page posted (see the Bottom sheet page).
// Uses Newtonsoft.Json (com.unity.nuget.newtonsoft-json) because `data` can be a string OR
an object.
using Newtonsoft.Json.Linq;

public void OnRawPageMessage(string json)
{
    JObject msg;
    try { msg = JObject.Parse(json); } catch { return; }
    var type = (string)msg["type"];
    var data = msg["data"];
    string token = data?.Type == JTokenType.String ? (string)data : (string)data?["token"];

    switch (type)
    {
        case "close": sheet.Close(); break;
        case "login":
            if (!string.IsNullOrEmpty(token)) StartCoroutine(ApplyNewLogin(token)); // POST
//api/platform/login
            else { IsLinked = true; StartCoroutine(Init(DeviceIdentity.Get()));
sheet.Close(); }
            break;
        case "token": break; // silent handshake - NOT a login
        case "logout":
            sheet.Close();
            PlayerPrefs.DeleteAll(); // forget identity, token and choice
            GameBoot.Restart(); // boot again as a new visitor
            break;
    }
}
```

Showing the balance

TypeScript (web)

```
// Inside BazinessPlatform: the player's Gita coin balance, straight from the platform. null
= unknown.
async getCoin(): Promise<number | null> {
    if (!this.enabled || !this.token || !this.apiBase) return null;
    try {
        const res = await fetch(`${this.apiBase}/user/get-coin`, {
            headers: { Authorization: `Bearer ${this.token}`, "Content-Type": "application/json" },
        });
        if (!res.ok) return null;
        const json = (await res.json()) as { amount?: number };
        return typeof json.amount === "number" ? json.amount : null;
    } catch {
        return null;
    }
}
```

```
// Wiring: refresh on every relevant moment.
baziness.onChange = () => void baziness.getCoin().then(renderCoinChip); // sheet closed,
login, link
window.addEventListener("focus", () => void baziness.getCoin().then(renderCoinChip));
```

React

```
// session.tsx – keep the external balance in context and refresh it on the right events.
const [gita, setGita] = useState<number | null>(null);

const refreshGita = useCallback(async () => {
  if (baziness.enabled) setGita(await baziness.getCoin());
}, []);

useEffect(() => {
  baziness.onChange = () => void refreshGita(); // fires when the sheet closes / login
changes
  void refreshGita();
  const onFocus = () => void refreshGita();
  window.addEventListener("focus", onFocus);
  return () => window.removeEventListener("focus", onFocus);
}, [refreshGita]);

// Anywhere in the UI:
<span className="gita-chip">
  <CurrencyIcon currency="GITA_COIN" />
  <bdi>{gita === null ? "..." : faNum(gita)}</bdi>
</span>
```

C# (Unity)

```
using System;
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;

[Serializable] class CoinResponse { public long amount; }

// apiBase and the user token come from /api/platform/token (see the Login page).
public IEnumerator GetCoin(Action<long?> done)
{
  if (!Enabled || string.IsNullOrEmpty(Token)) { done(null); yield break; }
  using var req = UnityWebRequest.Get($"{apiBase}/user/get-coin");
  req.SetRequestHeader("Authorization", $"Bearer {Token}");
  req.SetRequestHeader("Content-Type", "application/json");
  yield return req.SendWebRequest();
  if (req.result != UnityWebRequest.Result.Success) { done(null); yield break; }
  done(JsonUtility.FromJson<CoinResponse>(req.downloadHandler.text).amount);
}

// Refresh when the app regains focus and when the Baziness sheet closes.
void OnApplicationFocus(bool focus) { if (focus) StartCoroutine(GetCoin(UpdateCoinChip)); }
```

The reward button

TypeScript (web)

```
// Inside BazinessPlatform. GAME_NAME comes from /api/platform/config (or a constant).
openRewardSelection(): void {
  if (!this.enabled || !this.webBase) return;
  this.sheet.open(
    this.pageUrl("/shop", { redirect: "shop", extra: { game_name: this.gameName } }),
    BAZINESS_BOTTOM_BAR_CROP_PX, // optional: hide the platform's own tab bar (see below)
  );
}

// The lobby banner
rewardBanner.onclick = () => {
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود");
};
```

React

```
// Home.tsx – the «انتخاب جایزه» banner with the live balance.
const { gita } = useSession(); // from get-coin, see the Gita coins page

const openRewards = () => {
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود", "info");
};

return (
  <button className="hero-banner reward-banner" onClick={openRewards}>
    <span className="hero-text">
      <h2>انتخاب جایزه</h2>
      <span className="hero-sub">گیتا کوین‌هایت را با جایزه‌ی واقعی عوض کن</span>
      <span className="gita-chip">
        <CurrencyIcon currency="GITA_COIN" size={30} />
        <bdi className="num">{gita === null ? "..." : faNum(gita)}</bdi>
      </span>
    </span>
    <img className="hero-art" src={art("ui/banner-rewards")} alt="" />
  </button>
);
```

C# (Unity)

```
// RewardBanner.cs – attach to the lobby button.
public class RewardBanner : MonoBehaviour
{
  [SerializeField] BazinessLogin platform;
  [SerializeField] BazinessSheet sheet;
  [SerializeField] TMPro.TMP_Text balanceLabel;
  [SerializeField] string gameName = "example";

  void OnEnable() => StartCoroutine(platform.GetCoin(ShowBalance));

  public void OnClick()
  {
    if (!platform.Enabled) { Toast.Show("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود");
    return; }
    var url = platform.PageUrl("/shop", redirect: "shop", extra: $"game_name={
    Uri.EscapeDataString(gameName)}");
    sheet.Open(url, platform.OnRawPageMessage, onClose: () =>
```

```

StartCoroutine(platform.GetCoin(ShowBalance)));
}

void ShowBalance(long? amount) => balanceLabel.text = amount?.ToString("N0") ?? "...";
}

```

Hiding the bottom bar

The technique from [Reward page](#): make the frame taller than the view, and let the view clip it.

TypeScript (web)

```

/** Pixels cropped off the bottom of the embedded page. 0 = show the platform's tab bar. */
export const BAZINESS_BOTTOM_BAR_CROP_PX = 0;

// in BazinessSheet.open(url, cropBottomPx):
this.frame.style.marginBottom = cropBottomPx > 0 ? `${-cropBottomPx}px` : "";

```

Tutorial offer page

Open the offer page once during the tutorial, and give the starter pack when it closes ([Tutorial offer page](#)).

TypeScript (web)

```

// Inside BazinessPlatform.
gameOfferUrl = "";
private closeListeners = new Set<() => void>();

/** Run fn whenever the sheet closes; returns the unsubscribe function. */
onSheetClosed(fn: () => void) { this.closeListeners.add(fn); return () =>
this.closeListeners.delete(fn); }
// ...and in handleClosed(): for (const fn of [...this.closeListeners]) fn();

get canOpenSheet() { return this.enabled && !!this.webBase; }

/** Onboarding only. Adds the SDK parameters to the server-provided URL. */
openGameOffer() {
    if (!this.gameOfferUrl) return;
    const u = new URL(this.gameOfferUrl); // already has ?game_name=
    u.searchParams.set("packageName", this.packageName);
    u.searchParams.set("sdkType", "js"); // or its close button can't close
    the sheet
    if (this.token) u.searchParams.set("linkedToken", this.token);
    u.searchParams.set("allowGuest", "false");
    this.sheet.open(u.toString());
}

// Lobby: the coach-marked banner
function onRewardBannerTap() {
    if (onboarding.stage === "rewards") {
        onboarding.setStage("pack"); // persist first: a reload resumes at
        the pack
        if (baziness.canOpenSheet && baziness.gameOfferUrl) {

```

```

    const off = baziness.onSheetClosed(() => { off(); showStarterPack(); });
    baziness.openGameOffer();
  } else {
    showStarterPack(); // platform off: never block
  }
  onboarding
  }
  return;
}
baziness.openRewardSelection(); // after onboarding: the real shop
}

```

React

```

// Home.tsx: the onboarding branch of the «انتخاب جایزه» banner.
const openRewards = () => {
  if (coach === "rewards") {
    setOnboardingStage("pack"); // persisted in localStorage
    if (baziness.canOpenSheet && baziness.gameOfferUrl) {
      const off = baziness.onSheetClosed(() => {
        off();
        setCoach("pack"); // → StarterPackModal
      });
      setCoach("shop");
      baziness.openGameOffer();
    } else {
      setCoach("pack");
    }
    return;
  }
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازیس فعال میشود", "info");
};

{coach === "rewards" && (
  <div className="coach-veil" aria-hidden="true">
    <div className="coach-bubble">
      <img src={asset("/assets/ui/tutorial-guide.png")} alt="" />
      <p>!بزن تا ببینی با گیتا کوین‌هایت چه جایزه‌هایی می‌توانی بگیری </b></p>
    </div>
  </div>
)}

```

C# (Unity)

```

public void OnRewardBannerTap()
{
  if (Onboarding.Stage == OnboardingStage.Rewards)
  {
    Onboarding.SetStage(OnboardingStage.Pack); // persist before opening
    if (platform.Enabled && !string.IsNullOrEmpty(platform.GameOfferUrl))
    {
      var url = platform.GameOfferUrl
        + $"&packageName={Uri.EscapeDataString(platform.PackageName)}&sdkType=js"
        + $"&linkedToken={Uri.EscapeDataString(platform.Token ??
        "")}&allowGuest=false";
      sheet.Open(url, platform.OnRawPageMessage, onClosed: StarterPack.Show);
    }
    else StarterPack.Show(); // never block onboarding
    return;
  }
}

```

```

    }
    rewardBanner.OnClick();
}
// after onboarding: /shop

```

Purchase window

Open the tab **inside the tap**, start the purchase on your server, send the tab to the payment page, and verify when the player returns ([Baziness Pay](#)).

TypeScript (web)

```

// purchase.ts - the purchase popup.
export function buy(productId: string, ui: PurchaseUi) {
    const tab = window.open("about:blank", "_blank"); // 1. synchronously, no
    "noopener"
    let paymentId: string | null = null;
    let started = false;
    let verifying = false;

    const verify = async (manual: boolean) => {
        if (!started || verifying || !paymentId) return;
        verifying = true;
        ui.showChecking();
        try {
            const r = await api("POST", "/api/shop/purchase/verify", { paymentId });
            if (r.status === "FULFILLED") { cleanup(); ui.success(r.granted); void
refreshWallets(); return; }
            if (manual) ui.toast("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره بررسی کن.");
            ui.showWaiting();
        } catch (err) {
            if (manual) ui.error(err);
            ui.showWaiting();
        } finally { verifying = false; }
    };

    const onFocus = () => void verify(false);
    const onVisible = () => document.visibilityState === "visible" && void verify(false);
    const cleanup = () => { removeEventListener("focus", onFocus);
document.removeEventListener("visibilitychange", onVisible); };
    addEventListener("focus", onFocus);
    document.addEventListener("visibilitychange", onVisible);

    ui.showOpening(() => verify(true) /* «پرداخت کردم» */, () => { void verify(false); cleanup();
} /* close */);
    (async () => {
        try {
            const r = await api("POST", "/api/shop/purchase/start", { productId }); // 2.
            paymentId = r.paymentId;
            if (tab && !tab.closed) tab.location.href = r.url;
            else window.open(r.url, "_blank", "noopener");
            started = true;
            ui.showWaiting(); // 3.
        } catch (err) {
            tab?.close(); cleanup(); ui.error(err);
        }
    })();
}

```

```
})();
}
```

React

// Shop.tsx + PurchaseModal.tsx: open the tab in the click, then mount the modal.

```
export function openPayTab(): Window | null {
  return window.open("about:blank", "_blank"); // must run inside the click
  handler
}
```

// <button onClick={() => setBuying({ product, payTab: openPayTab() })}>...</button>

```
export function PurchaseModal({ product, payTab, onClose }: Props) {
  const { setWallets, toast, refresh } = useSession();
  const [phase, setPhase] = useState<"opening" | "waiting" | "checking" | "done">("opening");
  const paymentId = useRef<string | null>(null);
  const started = useRef(false);
  const verifying = useRef(false);

  const verify = useCallback(async (manual: boolean) => {
    if (!paymentId.current || verifying.current || !started.current) return;
    verifying.current = true;
    setPhase("checking");
    try {
      const res = await api<VerifyPurchaseResponse>("POST", "/api/shop/purchase/verify", {
        paymentId: paymentId.current });
      setWallets(res.wallets);
      if (res.status === "FULFILLED") { setPhase("done"); toast("خرید با موفقیت انجام شد",
        "success"); void refresh(); return; }
      if (manual) toast("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره بررسی کن",
        "info");
      setPhase("waiting");
    } catch (err) {
      if (manual) toast(errorMessage(err), "error");
      setPhase("waiting");
    } finally { verifying.current = false; }
  }, [refresh, setWallets, toast]);

  useEffect(() => { // start once
    (async () => {
      try {
        const res = await api<StartPurchaseResponse>("POST", "/api/shop/purchase/start", {
          productId: product.id });
        paymentId.current = res.paymentId;
        if (payTab && !payTab.closed) payTab.location.href = res.url;
        else window.open(res.url, "_blank", "noopener");
        started.current = true;
        setPhase("waiting");
      } catch (err) { payTab?.close(); toast(errorMessage(err), "error"); onClose(); }
    })();
  }, []);

  useEffect(() => { // verify when the player comes
    back
    const onFocus = () => void verify(false);
    const onVisible = () => document.visibilityState === "visible" && void verify(false);
    window.addEventListener("focus", onFocus);
    document.addEventListener("visibilitychange", onVisible);
  }, []);
}
```

```

        return () => { window.removeEventListener("focus", onFocus);
document.removeEventListener("visibilitychange", onVisible); };
    }, [verify]);

    const close = () => { if (phase !== "done" && paymentId.current) void verify(false);
onClose(); };

    return (
        <Modal open onClose={close}>
            <h3>{product.title}</h3>
            {phase === "done" ? <p>✓ به حساب اضافه شد</p>
                : phase === "waiting" ? (<
                    <p>صفحه‌ی پرداخت بازینس باز شد. بعد از پرداخت به بازی برگرد</p>
                    <button onClick={() => void verify(true)}>پرداخت کردم</button>
                </>)
                : <Spinner />}
        </Modal>
    );
}

```

C# (Unity)

```

// PurchaseFlow.cs – on mobile the payment page opens in the system browser, so there is no
// popup blocker. Verify when the app regains focus, plus a manual «پرداخت کردم» button.
public class PurchaseFlow : MonoBehaviour
{
    [SerializeField] GameApi api;           // your HTTP wrapper around /api/shop/purchase/*
    string paymentId;
    bool started, verifying;

    public IEnumerator Buy(string productId)
    {
        StartResponse r = null;
        yield return api.Post<StartResponse>("/api/shop/purchase/start", new { productId }, x
=> r = x);
        if (r == null) { Toast.Show("خطا در اتصال به درگاه"); yield break; }
        paymentId = r.paymentId;
        Application.OpenURL(r.url);         // {base}/buy?sku&uid&pname, built by YOUR server
        started = true;
        PurchasePanel.ShowWaiting(onPaid: () => StartCoroutine(Verify(manual: true)));
    }

    void OnApplicationFocus(bool focus)
    {
        if (focus && started) StartCoroutine(Verify(manual: false)); // back from the
browser
    }

    IEnumerator Verify(bool manual)
    {
        if (verifying || string.IsNullOrEmpty(paymentId)) yield break;
        verifying = true;
        VerifyResponse v = null;
        yield return api.Post<VerifyResponse>("/api/shop/purchase/verify", new { paymentId },
x => v = x);
        verifying = false;
        if (v?.status == "FULFILLED") { started = false; PurchasePanel.Success(v.granted);
Wallet.Set(v.wallets); }
        else if (manual) Toast.Show("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره"

```

```

    ". بررسی کن
    }

[System.Serializable] class StartResponse { public string paymentId, url; }
[System.Serializable] class VerifyResponse { public string status; public Grant[]
granted; public Wallets wallets; }
}

```

Cocos Creator

A complete set for Cocos Creator 3.x, in three files:

File	Role
BazinessSDK.ts	a persistent component: opens pages, routes their messages, reads the balance
BazinessWebView.ts	the component on your WebView prefab: loading overlay, close button, slide animations, messages
BazinessController.ts	boot wiring: <code>linkedToken</code> from the address, <code>/api/platform/token</code> , sign-in, sign-out, account switch

The prefab needs a `WebView`, a loading overlay node, a close button, and an `Animation` with `Open` and `Close` clips (slide up / slide down). On web builds the WebView is an `iframe`, so page messages arrive as `window message` events.

BazinessSDK.ts

```

// BazinessSDK.ts - a persistent component (put it on a node in the first scene).
// Opens Baziness pages in a WebView prefab, routes their messages and reads the Gita coin
balance.
import { _decorator, Component, director, find, instantiate, Prefab, EventTarget } from "cc";
import { BazinessWebView } from "../BazinessWebView";
const { ccclass, property } = _decorator;

export const bazinessEvents = new EventTarget(); // "login" | "logout" | "closed"

const WEB = { dev: "https://platform-web-dev.karizmastudios.org", prod:
"https://baziness.com" };
const API = { dev: "https://platform-dev.karizmastudios.org/api", prod:
"https://platform.karizmastudio.com/api" };

@ccclass("BazinessSDK")
export class BazinessSDK extends Component {
    @property(Prefab) webViewPrefab: Prefab = null!;

    static instance: BazinessSDK;
    static packageName = "com.yourstudio.yourgame";
    static devMode = true;
    static token: string | null = null; // the player's user token, from YOUR server

```

```

static isLinked = false;

private view: BazinessWebView | null = null;

onLoad() {
    BazinessSDK.instance = this;
    director.addPersistRootNode(this.node);
}

/** Call after your server answered /api/platform/token. */
static initialize(token: string, isLinked: boolean) {
    this.token = token;
    this.isLinked = isLinked;
}

static openLogin() {
    this.instance.open(`${this.web()}/`, { allowGuest: false });
}
static openProfile() {
    this.instance.open(`${this.web()}/profile`, { redirect: "profile", allowGuest: false });
}
static openRewards(gameName: string) {
    this.instance.open(`${this.web()}/shop`, { redirect: "shop", extra: { game_name: gameName
} });
}

/** The Gita coin balance, straight from the platform with the player's own token. */
static async getCoin(): Promise<number | null> {
    if (!this.token) return null;
    try {
        const res = await fetch(`${this.devMode ? API.dev : API.prod}/user/get-coin`, {
            headers: { Authorization: `Bearer ${this.token}`, "Content-Type": "application/json"
},
        });
        return res.ok ? ((await res.json()).amount ?? null) : null;
    } catch {
        return null;
    }
}

private static web() {
    return this.devMode ? WEB.dev : WEB.prod;
}

private open(base: string, o: { redirect?: string; allowGuest?: boolean; extra?:
Record<string, string> } = {}) {
    if (this.view) return; // one page at a time
    const q = new URLSearchParams(o.extra);
    q.set("packageName", BazinessSDK.packageName);
    q.set("sdkType", "js"); // the page posts close / login / logout / token
    if (BazinessSDK.token) q.set("linkedToken", BazinessSDK.token);
    if (o.redirect) q.set("redirect", o.redirect);
    if (o.allowGuest === false) q.set("allowGuest", "false");

    const node = instantiate(this.webViewPrefab);
    node.setParent(find("Canvas"));
    this.view = node.getComponent(BazinessWebView)!;
    this.view.onMessage = (m) => this.onMessage(m);
    this.view.onClosed = () => {
        this.view = null;
    }
}

```

```

        bazinessEvents.emit("closed");
    };
    this.view.open(`${base}?${q}`);
}

private onMessage(msg: { type: string; data?: unknown }) {
    const token = typeof msg.data === "string" ? msg.data : (msg.data as { token?: string } | undefined)?.token;
    switch (msg.type) {
        case "close":
            this.view?.close();
            break;
        case "login":
            if (token) {
                this.view?.close(true);
                bazinessEvents.emit("login", token); // → POST /api/platform/login on your server
            } else {
                BazinessSDK.isLinked = true; // linked in place
            }
            break;
        case "logout":
            this.view?.close(true);
            bazinessEvents.emit("logout");
            break;
        case "token":
            break; // the page's silent handshake – not a login
    }
}
}
}

```

BazinessWebView.ts

```

// BazinessWebView.ts – the component on the WebView prefab (WebView + loading overlay + close button).
import { _decorator, Animation, Button, Component, Node, WebView } from "cc";
const { ccclass, property } = _decorator;

@ccclass("BazinessWebView")
export class BazinessWebView extends Component {
    @property(WebView) webView: WebView = null!;
    @property(Node) loadingOverlay: Node = null!;
    @property(Button) closeButton: Button = null!;
    @property(Animation) animation: Animation = null!; // "Open" slides up, "Close" slides down

    onMessage: ((m: { type: string; data?: unknown }) => void) | null = null;
    onClosed: (() => void) | null = null;

    private origin = "";
    private closeTimer: ReturnType<typeof setTimeout> | null = null;
    private listener = (e: MessageEvent) => this.onWindowMessage(e);

    open(url: string) {
        this.origin = new URL(url).origin;
        this.webView.node.active = false;
        this.closeButton.node.active = false;
        // Our own close button appears after a few seconds: a way out even if the page never loads.
        this.closeTimer = setTimeout(() => {
            if (this.closeButton?.isValid) this.closeButton.node.active = true;
        });
    }
}

```

```

    }, 3000);
    this.closeButton.node.on(Button.EventType.CLICK, () => this.close(), this);
    this.webView.node.on(WebView.EventType.LOADED, () => {
        this.loadingOverlay.active = false;
        this.webView.node.active = true;
    }, this);
    this.webView.node.on(WebView.EventType.ERROR, () => this.close(), this);
    window.addEventListener("message", this.listener); // on web builds the WebView is an
iframe
    this.webView.url = url;
    this.animation.play("Open");
}

close(immediately = false) {
    window.removeEventListener("message", this.listener);
    if (this.closeTimer) clearTimeout(this.closeTimer);
    if (!immediately) this.animation.play("Close");
    setTimeout(() => {
        this.onClosed?.();
        if (this.node?.isValid) this.node.destroy();
    }, immediately ? 0 : 500);
}

private onWindowMessage(e: MessageEvent) {
    if (e.origin !== this.origin) return; // only the page we opened
    try {
        const msg = typeof e.data === "string" ? JSON.parse(e.data) : e.data;
        if (msg && typeof msg.type === "string") this.onMessage?.(msg);
    } catch {
        /* not a Baziness message */
    }
}
}
}

```

BazinessController.ts

```

// BazinessController.ts – boot-time wiring between your server, the SDK and the page
messages.
import { BazinessSDK, bazinessEvents } from "./BazinessSDK";
import { GameApi } from "./GameApi"; // your HTTP client for /api/*
import { LocalData } from "./LocalData"; // your localStorage wrapper

export class BazinessController {
    /** Returns true when it started an account switch or a reset – stop your own boot then. */
    static async initialize(): Promise<boolean> {
        // The Baziness app opens games with ?linkedToken=... . If it doesn't match the token this
device
        // already holds, a different Baziness account is arriving: start clean before going on.
        const fromUrl = new URLSearchParams(location.search).get("linkedToken");
        const saved = LocalData.get("baziness_token");
        if (fromUrl && saved && fromUrl !== saved) {
            this.resetAndReload();
            return true;
        }

        const resp = await GameApi.post("/api/platform/token", { linkedToken: fromUrl ?? saved
});
        if (!resp.enabled) return false; // platform off: no Baziness UI
        if (resp.switchToDeviceId) {

```

```

        this.switchTo(resp.switchToDeviceId);
        return true;
    }
    if (resp.token) {
        LocalData.set("baziness_token", resp.token);
        BazinessSDK.initialize(resp.token, resp.isLinkedToAccount);
    }

    bazinessEvents.on("login", async (token: string) => {
        const r = await GameApi.post("/api/platform/login", { token }); // the server validates
it
        if (r.switchToDeviceId) this.switchTo(r.switchToDeviceId);
        else if (r.ok) {
            LocalData.set("baziness_token", token);
            await this.initialize();
        }
    });
    bazinessEvents.on("logout", () => this.resetAndReload());
    return false;
}

private static switchTo(deviceId: string) {
    LocalData.clear();
    LocalData.set("device_id", deviceId);
    this.reload();
}

private static resetAndReload() {
    LocalData.clear(); // the next boot is a brand-new visitor
    this.reload();
}

private static reload() {
    const url = new URL(location.href);
    url.searchParams.delete("linkedToken");
    location.replace(url.toString()); // or reload your loading scene
}
}

```

Bottom sheet (in-game web view)

The swipe-down sheet that shows every Baziness page inside your game. It slides up, closes with a swipe, a tap outside or the page's own button, and passes the page's messages to your game.

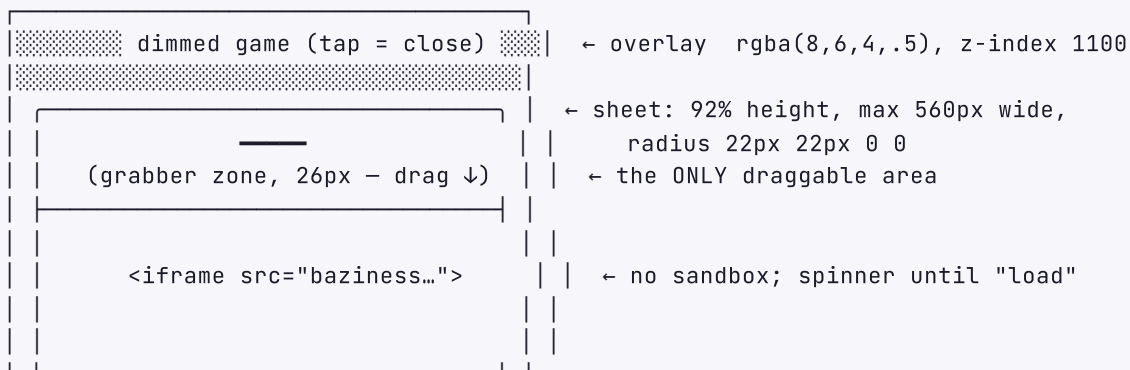
Why a sheet

Every Baziness page ([sign-in](#), [profile](#), [reward shop](#), [game offer](#)) opens **inside your game**, not in a new tab. The recommended container is a **bottom sheet**. It slides up over the dimmed game, stops a little short of the top, and has a rounded top edge with a drag handle. The player can close it in three ways, and the page can close it too.

Try it:

Anatomy

Text



How it must behave

1. **Opening:** draw the sheet below the screen, then animate it up on the next frame (about 320 ms, ease-out).
2. **Three ways to close:**
 - drag the **handle** down further than `min(150px, 28% of the sheet height)`; a shorter drag springs back
 - tap the **dimmed area** above the sheet
 - the page sends `{ "type": "close" }`

3. **Only the handle is draggable.** The page swallows its own touches, so a drag that starts inside the page can't move the sheet.
4. **No `sandbox` on the iframe.** The sign-in page needs its own cookies, pop-ups (for the code) and redirects. Allow `clipboard-read; clipboard-write; payment`.
5. **Show a spinner** until the page has loaded.
6. **Accept messages only from the page's own origin.** Read both string JSON and plain objects, and require a string `type`.
7. **Close cleanly:** animate down, then tear down **once**. Point the frame at `about:blank`, stop listening for messages, and only then report "closed". If the sheet was opened again in the meantime, do nothing.
8. **Stay above everything**, the splash screen included: the sign-in page can open from the splash.
9. **Size it like your game's column:** full width on phones, at most about 560 px wide and centred on wide screens.

Web: TypeScript, React and CSS

TypeScript (web)

```
// bazinesswebview.ts - the sheet, framework-free.
export interface BazinessMessage { type: string; data?: unknown }

export class BazinessSheet {
  readonly el: HTMLElement;
  private sheet: HTMLElement;
  private frame: HTMLIFrameElement;
  private spinner: HTMLElement;
  private allowedOrigin: string | null = null;
  private isOpen = false;
  private dragging = false;
  private dragStartY = 0;
  private dragDelta = 0;
  private sheetHeight = 1;

  onMessage: ((m: BazinessMessage) => void) | null = null;
  onClosed: (() => void) | null = null;

  constructor() {
    this.frame = Object.assign(document.createElement("iframe"), { className: "baziness-frame" });
    this.frame.setAttribute("allow", "clipboard-read; clipboard-write; payment"); // no
    sandbox!
    this.spinner = Object.assign(document.createElement("div"), { className: "baziness-spinner" });
    const grabberZone = document.createElement("div");
    grabberZone.className = "baziness-grabber-zone";
    grabberZone.innerHTML = '<div class="baziness-grabber"></div>';
    this.sheet = document.createElement("div");
    this.sheet.className = "baziness-sheet";
    this.sheet.append(grabberZone, this.spinner, this.frame);
    this.el = document.createElement("div");
```

```

this.el.className = "overlay baziness-webview hidden";
this.el.append(this.sheet);
document.body.append(this.el);

this.frame.addEventListener("load", () => this.spinner.classList.add("hidden"));
grabberZone.addEventListener("pointerdown", (e) => this.dragStart(e));
grabberZone.addEventListener("pointermove", (e) => this.dragMove(e));
grabberZone.addEventListener("pointerup", () => this.dragEnd());
grabberZone.addEventListener("pointercancel", () => this.dragEnd());
this.el.addEventListener("pointerdown", (e) => { if (e.target === this.el) this.close();
}); // tap outside
}

open(url: string, cropBottomPx = 0): void {
  if (this.isOpen) return;
  this.isOpen = true;
  this.frame.style.marginBottom = cropBottomPx > 0 ? `${-cropBottomPx}px` : "";
  this.allowedOrigin = new URL(url).origin;
  this.spinner.classList.remove("hidden");
  this.el.classList.remove("hidden");
  this.sheet.style.transition = "none";
  this.sheet.style.transform = "translateY(100%)";
  this.frame.src = url;
  window.addEventListener("message", this.onWindowMessage);
  requestAnimationFrame(() => {
    this.sheet.style.transition = "";
    this.sheet.style.transform = "translateY(0)";
  });
}

close(): void {
  if (!this.isOpen) return;
  this.isOpen = false;
  this.dragging = false;
  window.removeEventListener("message", this.onWindowMessage);
  this.sheet.style.transition = "";
  this.sheet.style.transform = "translateY(100%)";
  let torn = false;
  const teardown = () => {
    if (torn || this.isOpen) return; // already torn down, or reopened meanwhile
    torn = true;
    this.sheet.removeEventListener("transitionend", teardown);
    this.el.classList.add("hidden");
    this.frame.src = "about:blank";
    this.frame.style.marginBottom = "";
    this.sheet.style.transform = "";
    this.allowedOrigin = null;
    this.onClosed?();
  };
  this.sheet.addEventListener("transitionend", teardown);
  window.setTimeout(teardown, 400);
}

private dragStart(e: PointerEvent) {
  this.dragging = true;
  this.dragStartY = e.clientY;
  this.dragDelta = 0;
  this.sheetHeight = this.sheet.getBoundingClientRect().height || 1;
  this.sheet.style.transition = "none";
  (e.currentTarget as HTMLElement).setPointerCapture(e.pointerId);
}

```

```

}
private dragMove(e: PointerEvent) {
  if (!this.dragging) return;
  this.dragDelta = Math.max(0, e.clientY - this.dragStartY); // downward only
  this.sheet.style.transform = `translateY(${this.dragDelta}px)`;
}
private dragEnd() {
  if (!this.dragging) return;
  this.dragging = false;
  this.sheet.style.transition = "";
  if (this.dragDelta > Math.min(150, this.sheetHeight * 0.28)) this.close();
  else this.sheet.style.transform = "translateY(0)"; // spring back
}

private onWindowMessage = (event: MessageEvent) => {
  if (this.allowedOrigin && event.origin !== this.allowedOrigin) return;
  let value: unknown = event.data;
  if (typeof value === "string") { try { value = JSON.parse(value); } catch { return; } }
  if (value && typeof value === "object" && typeof (value as BazinessMessage).type ===
"string") {
    this.onMessage?.(value as BazinessMessage);
  }
};
}

```

React

```

// BazinessSheet.tsx – the React version. Mount once at the app root; baziness.host drives
it.
import { useEffect, useRef, useState } from "react";
import { createPortal } from "react-dom";
import { baziness } from "../platform/baziness";

export function BazinessSheet() {
  const [url, setUrl] = useState<string | null>(null);
  const [shown, setShown] = useState(false);
  const [loading, setLoading] = useState(true);
  const [drag, setDrag] = useState(0);
  const dragStart = useRef<number | null>(null);

  useEffect(() => {
    baziness.host = {
      open: (next) => {
        setLoading(true);
        setUrl(next);
        requestAnimationFrame(() => requestAnimationFrame(() => setShown(true))); // slide up
      },
      close: () => setShown(false),
    };
    return () => { baziness.host = null; };
  }, []);

  useEffect(() => { // messages, from the page's origin
    only
    if (!url) return;
    const origin = new URL(url).origin;
    const onMessage = (e: MessageEvent) => {
      if (e.origin !== origin) return;
      const msg = parse(e.data); // see "Handling page messages"
    };
  }, [url]);
}

```

```

(Client code)
    if (msg) baziness.handleMessage(msg);
};
window.addEventListener("message", onMessage);
return () => window.removeEventListener("message", onMessage);
}, [url]);

const onTransitionEnd = () => {
    if (!shown && url) {                                     // finished sliding down → tear down
once
        setUrl(null);
        setDrag(0);
        baziness.handleClosed();
        measureViewport();                                // the phone may have moved its nav
bar
    }
};

if (!url) return null;
return createPortal(
    <div className={`overlay baziness-overlay ${shown ? "is-open" : ""}` `
        onPointerDown={(e) => e.target === e.currentTarget && setShown(false)}>
        <div className="baziness-sheet"
            style={{ transform: shown ? `translateY(${drag}px)` : "translateY(100%)",
                transition: dragStart.current !== null ? "none" : undefined }}
            onTransitionEnd={onTransitionEnd}>
            <div className="baziness-grabber-zone"
                onPointerDown={(e) => { dragStart.current = e.clientY;
e.currentTarget.setPointerCapture(e.pointerId); }}
                onPointerMove={(e) => dragStart.current !== null && setDrag(Math.max(0,
e.clientY - dragStart.current))}
                onPointerUp={() => { dragStart.current = null; if (drag > 140) setShown(false);
else setDrag(0); }}>
                <span className="baziness-grabber" />
            </div>
            {loading && <div className="baziness-loading"><Spinner /></div>}
            <iframe className="baziness-frame" src={url} title="بازينس"
                allow="clipboard-read; clipboard-write; payment" onLoad={() =>
setLoading(false)} />
            </div>
        </div>,
        document.getElementById("app-frame") ?? document.body,
    );
}

```

CSS

```

/* Above the splash (z 1000) so it shows during the first-time login choice. */
.overlay.baziness-webview {
    position: fixed;
    inset: 0;
    z-index: 1100;
    display: flex;
    justify-content: center;
    align-items: flex-end;          /* dock the sheet to the bottom */
    background: rgba(8, 6, 4, 0.5); /* the dimmed, tap-to-close area */
}
.overlay.baziness-webview.hidden { display: none; }

```

```

.baziness-sheet {
  position: relative;
  width: min(560px, 100%);           /* full width on phones, capped on wide screens */
  height: 92%;                       /* stops short of the top */
  display: flex;
  flex-direction: column;
  background: #fff;
  border-radius: 22px 22px 0 0;
  overflow: hidden;                  /* also clips the cropped bottom bar */
  box-shadow: 0 -10px 34px rgba(0, 0, 0, 0.42);
  transform: translateY(100%);
  transition: transform 320ms cubic-bezier(0.22, 0.61, 0.36, 1);
  will-change: transform;
}

/* The only draggable area – the iframe captures its own pointer events. */
.baziness-grabber-zone {
  flex: 0 0 auto;
  height: 26px;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: grab;
  touch-action: none;
  background: #fff;
}

.baziness-grabber {
  width: 44px;
  height: 5px;
  border-radius: 999px;
  background: rgba(0, 0, 0, 0.22);
}

.baziness-frame {
  flex: 1 1 auto;
  width: 100%;
  min-height: 0;
  border: 0;
  background: #fff;
}

.baziness-spinner {
  position: absolute;
  top: 50%;
  left: 50%;
  width: 42px;
  height: 42px;
  margin: -21px 0 0 -21px;
  border-radius: 50%;
  border: 4px solid rgba(120, 90, 46, 0.25);
  border-top-color: #b4824a;
  animation: spin 0.9s linear infinite;
}

.baziness-spinner.hidden { display: none; }
@keyframes spin { to { transform: rotate(360deg); } }

```

Cocos Creator

The `BazinessWebView` component in [Client code → Cocos Creator](#) does the same job with a `WebView` prefab: a loading overlay until the page loads, a close button that appears after a few seconds, and `Open` / `Close` slide animations.

Unity

Unity has no built-in web view, so you need a plugin. This sketch uses [gree/unity-webview](#). The page starts 8% below the top of the screen, the drag handle is a Unity UI element above it, and the page's messages reach C# through a small JavaScript bridge.

C# (Unity)

```
// BazinessSheet.cs - a bottom sheet around a WebView plugin (gree/unity-webview).
using System;
using UnityEngine;
using UnityEngine.EventSystems;

public class BazinessSheet : MonoBehaviour, IDragHandler, IEndDragHandler // put this on
the grabber UI
{
    [SerializeField] RectTransform grabber; // a 26px bar drawn by Unity above the web
view
    [SerializeField] GameObject dimmer; // full-screen button behind; OnClick →
Close()
    WebViewObject web;
    Action<string> onMessage;
    Action onClosed;
    float topMargin, drag;

    public void Open(string url, Action<string> onMessage, Action onClosed = null)
    {
        this.onMessage = onMessage;
        this.onClosed = onClosed;
        topMargin = Screen.height * 0.08f; // the sheet stops 8% short of the top
        drag = 0;
        web = new GameObject("BazinessWebView").AddComponent<WebViewObject>();
        web.Init(
            cb: msg => this.onMessage?.Invoke(msg), // messages forwarded by the bridge
            below
            ld: _ => web.EvaluateJS(
                "window.addEventListener('message',function(e){" +
                "Unity.call(typeof e.data=='string'?e.data:JSON.stringify(e.data));});"));
        Layout();
        web.LoadURL(url);
        web.SetVisibility(true);
        dimmer.SetActive(true);
        grabber.gameObject.SetActive(true);
    }

    public void OnDrag(PointerEventData e) { drag = Mathf.Max(0, drag - e.delta.y); Layout();
}

    public void OnEndDrag(PointerEventData e)
```

```

{
    if (drag > Mathf.Min(150, (Screen.height - topMargin) * 0.28f)) Close();
    else { drag = 0; Layout(); } // spring back
}

public void Close()
{
    if (web == null) return;
    Destroy(web.gameObject);
    web = null;
    dimmer.SetActive(false);
    grabber.gameObject.SetActive(false);
    onClosed?.Invoke();
}

void Layout()
{
    int top = (int)(topMargin + drag + grabber.rect.height);
    web.SetMargins(0, top, 0, 0);
    grabber.anchoredPosition = new Vector2(0, -(topMargin + drag));
}
}

```

Common mistakes

Mistake	What happens
<code>sandbox</code> on the iframe	sign-in gets stuck: pop-ups and cookies are blocked
The whole sheet is draggable	drags inside the page scroll it and move the sheet, or nothing moves
No origin check on messages	a message from somewhere else is taken for a page message
"Closed" reported twice (animation end + timeout)	a waiting sign-in finishes twice; the starter pack opens twice
The sheet sits below the splash screen	the sign-in page opens invisibly behind it
The frame isn't reset to <code>about:blank</code>	the old page keeps running in the background

Example game — Rock, Paper, Scissors

A complete, small game that uses every integration in this wiki. Play it on the dev server, download the project, and run it yourself in two minutes.

Play it

[Open the example game](#) (development servers: platform-dev, platform-web-dev, pay-dev).

[Download the project \(.zip\)](#): plain Node.js and browser JavaScript, **no dependencies**, no build step.

What it shows

A best-of-five rock-paper-scissors match against a bot: first to **3 points** wins, and draws don't count. Each match costs **1 ticket**, and winning pays **50 Gita coins**. Every platform feature is in it:

Feature	Where in the game	Wiki page	Code
Sign in with Baziness (splash choice), account switch, logout	first launch; Settings → «اتصال حساب»	Login	<code>public/js/platform. server/services.js (platformToken , platformLogin)</code>
Currencies: tickets (local) and Gita coins (<code>get-coin</code>)	the top bar chips, the reward banner	Gita coins	<code>public/js/platform. (getCoin), public/js/app.js (renderTopbar)</code>
Winner gets 50 Gita coins (<code>add-coin</code> , idempotent), sign-in required to claim	the result popup, the "reward waiting" strip	Gita coins	<code>server/services.js (claimMatchReward), public/js/app.js (claim)</code>
«انتخاب جایزه» reward page	the purple banner	Reward page	<code>public/js/platform. (openRewardSelection</code>
Tutorial: rules → coach mark → game-offer page → «خرید موفق» starter pack (5 tickets)	first launch	Tutorial offer page	<code>public/js/app.js (tutorial section), server/services.js (claimStarterPack)</code>
Swipe-down sheet	every Baziness page	Bottom sheet	<code>public/js/sheet.js , public/styles.css</code>
Baziness Pay: «بسته ی بلیت», 10 tickets for 4,900 Toman (SKU <code>example_pack</code>)	the shop card	Baziness Pay	<code>server/pay.js , server/services.js (startPurchase , verifyPurchase), public/js/purchase.</code>

Feature	Where in the game	Wiki page	Code
UTM / <code>?tag=</code> first-touch capture	any link, e.g. <code>.../example/?utm_source=wiki&utm_medium=docs</code>	Links & UTM	<code>public/js/api.js</code> (<code>FIRST_TOUCH</code>), <code>server/services.js</code> (<code>ensureUser</code>)

Run it locally

Shell

```
unzip baziness-example-rps.zip && cd baziness-example-rps
cp .env.example .env          # fill in BAZINESS_CLIENT_ID / BAZINESS_CLIENT_SECRET
npm start                     # → http://127.0.0.1:4380/
```

Requirements: **Node.js 20+**. Data lives in `data/db.json`, a tiny JSON store standing in for your database.

<code>.env</code>	Development value
<code>BAZINESS_CLIENT_ID</code> / <code>BAZINESS_CLIENT_SECRET</code>	the example's dev client (ask the platform team, or use your own game's)
<code>BAZINESS_PLATFORM_PACKAGE_NAME</code>	<code>com.karizmastudio.example</code>
<code>BAZINESS_PAY_BASE_URL</code>	<code>https://pay-dev.karizmastudios.org</code>
<code>BAZINESS_PAY_PACKAGE_NAME</code>	<code>com.karizmastudio.example</code>
<code>PACK_SKU</code> / <code>PACK_PRICE</code>	<code>example_pack</code> / <code>4900</code>

Tip

No credentials? It still runs. With `BAZINESS_PLATFORM_ENABLED=false` the Baziness UI disappears and winnings are kept locally. With `DEV_FAKE_BAZINESS_PAY=true` and `BAZINESS_PAY_BASE_URL=http://localhost:4380/dev/baziness`, purchases go through a built-in fake gateway with a «پرداخت موفق» button.

Testing tips

- **Signing in:** on this dev deployment, «ورود با حساب بازیکن» accepts **any phone number with the code** `111111`. No SMS is sent.

- **Replay the tutorial:** use «بازنشانی این دستگاه» in the lobby footer. It clears local storage, so the next boot is a brand-new player.
- **A second player in the same browser:** add `?device=someone-else` to the URL.
- **Claiming as a guest:** win a match without signing in. The claim answers `PLATFORM_NOT_LINKED` and the login sheet opens. After you sign in, the claim retries by itself.
- **A purchase you finish later:** start a purchase, pay, close the game, and open it again. The lobby reconciles pending payments.
- **Campaign links:** open `.../example/?utm_source=wiki&utm_campaign=test`. The address is clean after load, and the tags are stored on the account when it is created.

Project layout

Text

```

baziness-example-rps/
├── .env.example           every setting, documented
├── package.json          "npm start" - no dependencies
├── server/
│   ├── index.js          HTTP routes + static files (relative paths: works at / or
/example/)
│   ├── env.js            configuration (.env loader)
│   ├── platform.js       platform client: auth, create-user, tokens, add-coin ← secret
└── stays here
    ├── pay.js            Baziness Pay: buy URL, get_payment, verify_payment (form), consume
    ├── services.js       accounts & linking, match rules, rewards, starter pack, purchases
    ├── store.js          JSON-file store (swap for your database)
    └── fake-pay.js       local fake gateway (never in production)
├── public/
│   ├── index.html, styles.css
│   ├── assets/          art (hands, bot, pack, prize, Gita coin, ticket, Baziness logo)
│   └── js/
│       ├── api.js        first module: UTM capture, device id, fetch wrapper
│       ├── platform.js   BazinessPlatform: page URLs, messages, login, get-coin
│       ├── sheet.js      the swipe-down sheet
│       ├── purchase.js   Pay popup: tab opened in the click, verify on focus
│       ├── ui.js         DOM helpers, Persian numbers, toasts, modals
│       └── app.js        splash, lobby, match, tutorial

```

Hosting it under a path

The client only uses **relative** URLs (`api/...`, `assets/...`), so the same files work at `/` or under a prefix. On the wiki's dev server, nginx strips `/example/` and proxies to the Node process:

NGINX

```

location = /example { return 301 /example/; }
location /example/ {
    proxy_pass http://127.0.0.1:4380/;           # trailing slash: /example/api/me → /api/me
}

```

```
proxy_set_header Host $host;  
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
}
```

Go-live checklist & troubleshooting

Everything to check before you switch Baziness on in production, and the fixes for the problems integrations run into most often.

Before you go live

Settings

- ☐ A **production** client id and secret (different from development), on the server only.
- ☐ `BAZINESS_PLATFORM_ENABLED=true`, `BAZINESS_PLATFORM_USE_PRODUCTION=true`, and the platform package name and game name set.
- ☐ `BAZINESS_PAY_ENABLED=true`, `BAZINESS_PAY_BASE_URL=https://pay.baziness.com`, and the Pay package name set; the local fake gateway off.
- ☐ Search your client build for the secret: it must **not** be there.
- ☐ `/api/platform/config` answers `enabled: true` with the **production** addresses.

Login

On the dev platform you can test every case with made-up phone numbers and the code `111111`, because no SMS is sent ([details](#)).

- ☐ The splash asks «ورود با حساب بازینس» / «ادامه به عنوان مهمان» once per device.
- ☐ Closing the sign-in page without signing in brings the two buttons back.
- ☐ Signing in to an account used on another device **switches** to it, with its progress.
- ☐ Settings → connect account opens the profile page, and shows "connected" afterwards.
- ☐ Signing out inside a Baziness page resets the game to a brand-new visitor.
- ☐ Opening the game from the Baziness app (`?linkedToken=`) still runs the tutorial for new players.

Coins & pages

- ☐ The Gita coin balance comes from the platform: `...` while unknown, never a made-up `0`.
- ☐ The balance refreshes when a page closes and when the game regains focus.
- ☐ Every one-off payout uses a `transactionId` built from what the payout is, and a `400` counts as paid.
- ☐ «انتخاب جایزه» opens `/shop?game_name=...` without a sign-in wall.
- ☐ (If you use it) the tutorial's first tap opens `/game-offer`, and closing it continues to the starter pack.

- ☐ The view closes by its handle, by a tap outside, and by the page's own button.

Links

- ☐ Every page link has `packageName` and `sdkType=js`, and `linkedToken` when available.
- ☐ Campaign links carry all four `utm_*` in ASCII, and the address bar is clean after loading.
- ☐ The campaign survives a Baziness sign-in and an account switch.

Baziness Pay

- ☐ Every SKU is registered under your Pay package name.
- ☐ A real production purchase is given **once**, then consumed.
- ☐ Buying the same pack twice in a row works, which proves `consume` ran.
- ☐ A payment finished after closing the game is given the next time the shop opens.

Troubleshooting

Symptom	Likely cause	Fix
The page's close button does nothing	<code>sdkType=js</code> missing, or the message origin check uses the wrong origin	add <code>sdkType=js</code> to every page link; compare the message origin with the address you opened
The reward shop closes as soon as it opens	the <code>token</code> handshake is treated as a sign-in	ignore <code>token</code> ; only <code>login</code> is a sign-in
The first sign-in hangs, or your server answers 400	the sign-in started before the player had a Baziness user and token	call <code>/api/platform/token</code> first, and open the page with that token
Signed in, but the game still shows a guest	<code>/api/platform/login</code> wasn't called, or the profile wasn't reloaded	call it on <code>login</code> , then reload the player's profile
The sign-in page offers "continue as guest"	<code>allowGuest=false</code> missing	add it on the sign-in, profile and offer pages (not <code>/shop</code>)
The reward page asks the player to sign in	the site root was opened instead of <code>/shop</code>	open <code>{web}/shop?game_name=...</code> with <code>linkedToken</code>
Platform calls start failing after a while	the client token expired (<code>401</code>)	get a new one (step 1) and retry once
A valid-looking token is rejected	<code>validate-user-token</code> answered <code>isValid: false</code> (HTTP 200)	check the <code>isValid</code> field, not only the HTTP status; the client signs in again
A player got the same coins twice	the <code>transactionId</code> was built from the time	build it from what the payout is (step 5)
A claim fails forever after a crash	<code>add-coin</code> answered <code>400</code> and the claim treated it as an error	treat <code>400</code> as "already paid"
The payment tab stays blank	the tab was opened after waiting for the server, or with <code>noopener</code>	open an empty tab inside the tap, then set its address

Symptom	Likely cause	Fix
The player paid but got nothing	they never came back, so the check never ran	check on focus and on پرداخت «کردم», and check pending payments when the shop opens
<code>verify_payment</code> always fails	it was sent as JSON	send a form body
Pay answers <code>no_required_params</code>	<code>uid</code> isn't numeric	send your numeric player id
A second purchase of the same pack is blocked	the purchase wasn't consumed	call <code>consume</code> after giving the item, with retries
Players from the Baziness app skip the tutorial	<code>?linkedToken=</code> was taken to mean "returning player"	decide "new player" from your own data
The campaign is lost after sign-in	the tags were read from the address only, and the game restarted	store them on the device before the sign-in
Persian campaign names are missing	UTM values are kept as ASCII only	use ASCII names (<code>shahrivar_1405</code>)
The sign-in view is invisible on the splash	the view sits below the splash screen	put it above everything

Getting help

- Credentials, your game's offer page and SKUs: ask the **Baziness platform team**.
- To see everything working together, read the [example game](#)'s code.

API reference

Every endpoint, address and message on one page — the platform API, Baziness Pay, the Baziness pages, and the endpoints your own server should offer.

Platform API

- **Production:** <https://platform.karizmastudio.com/api>
- **Development:** <https://platform-dev.karizmastudios.org/api>

Step-by-step explanations: [Platform API](#).

Method & path	Auth	Body	Response	Notes
POST <code>/client/auth</code>	—	<code>{ clientId, clientSecret }</code>	<code>{ clientId, token }</code>	the client token
POST <code>/client/create-user</code>	client token	<code>{ deviceId, platform, store, linkedToken }</code>	<code>{ userId, token, isLinkedToAccount }</code>	<code>platform: Web / Android / Ios / Editor ; store: Direct / GooglePlay / AppStore / CafeBazaar / Myket</code>
GET <code>/client/get-user-token/{userId}</code>	client token	—	<code>{ userId, token, isLinkedToAccount }</code>	a fresh user token
GET <code>/client/validate-user-token/{token}</code>	client token	—	<code>{ userId, isValid }</code>	invalid → 200 with <code>isValid: false</code> , <code>userId: 0</code>
POST <code>/client/add-coin</code>	client token	<code>{ userId, amount, transactionId, metaData }</code>	<code>{ resourceId, amount }</code> (new balance)	a repeated <code>transactionId</code> → 400 { <code>success: false, error: "add resource failed"</code> }
GET <code>/user/get-coin</code>	user token	—	<code>{ resourceId, amount }</code>	called by the client; any origin allowed

Any call with a missing or expired client token answers 401 .

Baziness pages

- **Production:** <https://baziness.com>
- **Development:** <https://platform-web-dev.karizmastudios.org>

Details: [Baziness pages](#).

Page	Path	Required	Optional
Sign-in	/	<code>packageName</code> , <code>sdkType=js</code>	<code>linkedToken</code> , <code>allowGuest=false</code> , <code>buildStore</code>
Profile	/profile	<code>packageName</code> , <code>sdkType=js</code>	<code>linkedToken</code> , <code>redirect=profile</code> , <code>allowGuest=false</code>
Reward shop «انتخاب جایزه»	/shop	<code>packageName</code> , <code>sdkType=js</code> , <code>linkedToken</code>	<code>game_name</code> , <code>redirect=shop</code>
Game offer	/game- offer	<code>game_name</code> , <code>packageName</code> , <code>sdkType=js</code>	<code>linkedToken</code> , <code>allowGuest=false</code>

Messages from the pages

`{ "type": string, "data"? : string | { "token": string } }` , as JSON text or an object.

type	What to do
<code>login</code> with a token	send it to your server → verify → link or switch → close
<code>login</code> without data	linked in place → refresh
<code>token</code>	ignore (a silent handshake)
<code>logout</code>	close, clear local data, restart as a new visitor
<code>close</code>	close the view

Baziness Pay

- **Production:** <https://pay.baziness.com>
- **Development:** <https://pay-dev.karizmastudios.org>

`uid` must be numeric. Details: [Baziness Pay](#).

Method & path	Body format	Body	Response
GET /buy	query	sku , uid , pname	the payment page
POST /get_payment	JSON	{ uid, sku, pname }	{ uid, sku, token, time, success } , or { error: "payment_is_not_valid", success: false }
POST /verify_payment	form	uid , token , sku , pname	{ success, error }
POST /consume	JSON	{ uid, sku, pname }	{ success }

Your server's endpoints (recommended)

Endpoint	Body	Response
GET /api/platform/config	—	{ enabled, webBase, apiBase, packageName, gameName, gameOfferUrl }
POST /api/platform/token	{ deviceId, linkedToken?, tag?, utm? }	{ enabled, ...settings, token?, userId?, isLinkedToAccount?, switchToDeviceId?, error? }
POST /api/platform/login	{ deviceId, token }	{ ok, switchToDeviceId?, reason? }
GET /api/shop	—	the products this player can buy
POST /api/shop/purchase/start	{ productId }	{ paymentId, url }
POST /api/shop/purchase/verify	{ paymentId }	{ status, granted, balances }
POST /api/shop/purchase/cancel	{ paymentId }	—
GET /api/shop/purchase/pending	—	[{ paymentId, productId }]

Settings

Setting	Default	Meaning
BAZINESS_PLATFORM_ENABLED	false	master switch for accounts, coins and pages
BAZINESS_PLATFORM_USE_PRODUCTION	false	production or development addresses
BAZINESS_CLIENT_ID / BAZINESS_CLIENT_SECRET	—	credentials (secret)
BAZINESS_PLATFORM_PACKAGE_NAME	—	packageName on page links
BAZINESS_GAME_NAME	—	game_name on the reward and offer pages
BAZINESS_PLATFORM_API_URL / BAZINESS_PLATFORM_WEB_URL	derived	address overrides
BAZINESS_GAME_OFFER_URL	{web}/game-offer? game_name=...	offer page override
BAZINESS_PAY_ENABLED	false	master switch for purchases
BAZINESS_PAY_BASE_URL	—	https://pay-dev.karizmastudios.org or https://pay.baziness.com
BAZINESS_PAY_PACKAGE_NAME	—	pname
BAZINESS_SKU_PREFIX	empty	SKU = prefix + product id
DEV_FAKE_BAZINESS_PAY	false	the local fake gateway (never in production)