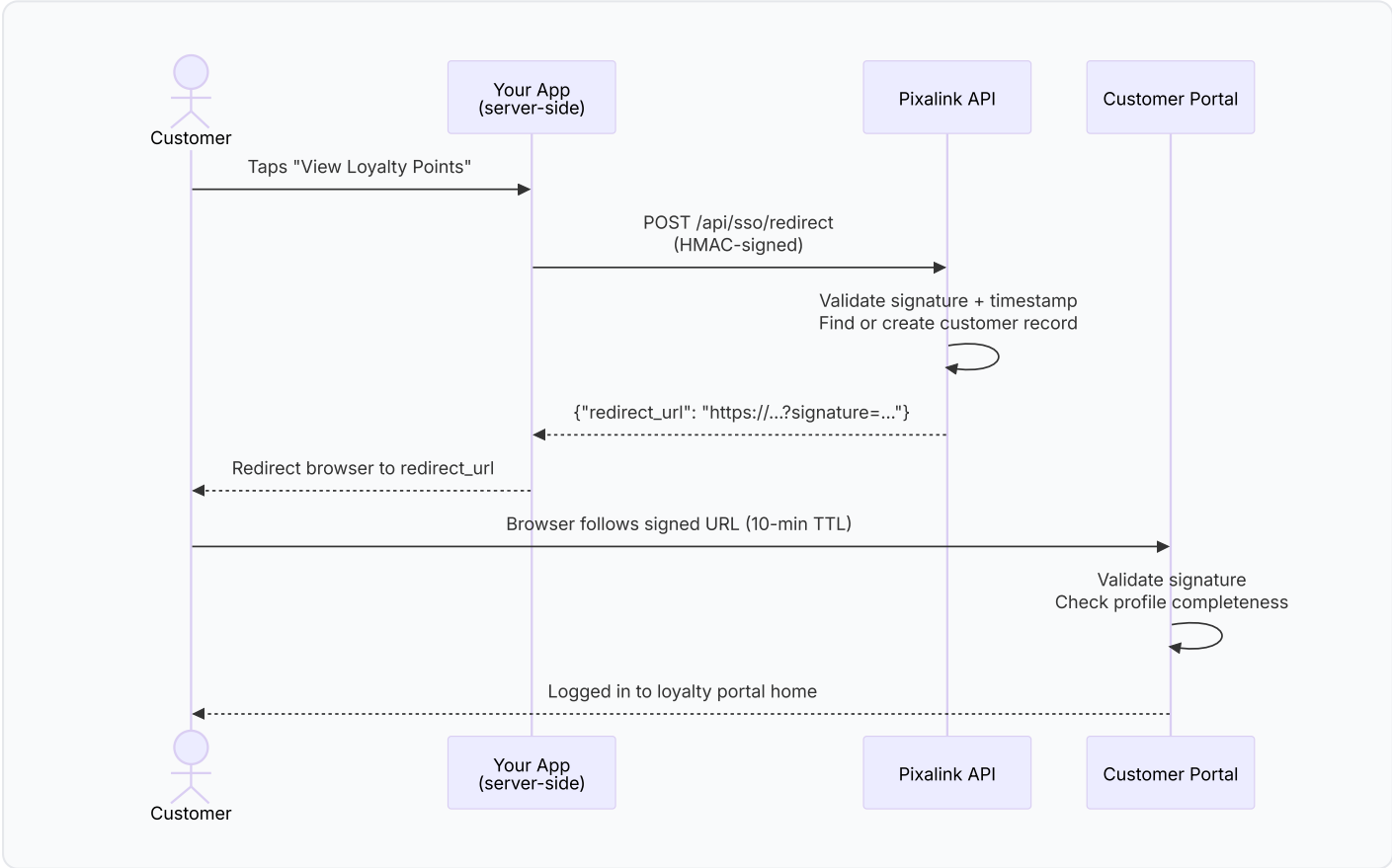
What Is This?

The Pixalink Customer Portal SSO lets your app log customers directly into their Pixalink loyalty portal — no separate login needed. Your backend calls a signed API endpoint; Pixalink returns a short-lived URL; you redirect the customer's browser there.

This is designed for apps that already have authenticated users (mobile apps, web portals, POS kiosks) and want a seamless handoff into the Pixalink loyalty experience.

> **This is not the same as the vendor OAuth2 SSO.** The OAuth2 SSO guide covers logging vendor (admin panel) users into your microservice via Passport. This guide is for logging **end customers** into the **customer loyalty portal**.

How It Works



Steps 2–5 are server-side and invisible to the customer. The redirect happens in under a second.

Prerequisites

Before integrating, ensure you have:

- **SSO client credentials** — a `client_id` (UUID) and `client_secret` provisioned by Pixalink support. The client type must be `sso` (distinct from standard OAuth2 clients).
- **OAuth2 feature enabled** on your organisation — contact Pixalink support.
- The customer's **phone number in E.164 format** (e.g. `+60123456789`) — this is the primary customer identifier in Pixalink.
- A valid `space_id` for your organisation. Each organisation has one or more spaces (outlets/branches). Ask your Pixalink account manager for your space IDs.

The Endpoint

```
POST /api/sso/redirect
```

API reference: POST /api/sso/redirect (https://explore.pixalink.io/docs#sso-POSTapi-sso-redirect) — full schema, try-it-out, and live response examples.

Rate limit: 30 requests per minute per client.

Headers

Header	Required	Value
`X-Client-Id`	Yes	Your SSO client UUID
`X-Signature`	Yes	`HMAC-SHA256(raw_body, client_secret)` in hex
`Content-Type`	Yes	`application/json`

Request Body

```
{
  "customer_phone": "+60123456789",
  "space_id": 42,
  "timestamp": 1746518400,
  "redirect_to": "rewards"
}
```

Field	Type	Description
`customer_phone`	string	E.164 phone number — must start with `+`
`space_id`	integer	Pixalink space ID — must belong to your organisation
`timestamp`	integer	Unix timestamp (seconds). Must be within ±60s of server time
`redirect_to`	string	**Optional.** Page to land on after login. Defaults to `home`. Allowed: `home`, `profile`, `rewards`, `points`, `credits`, `membership`, `order`, `reservations`, `referral`

Success Response — `200 OK`

```
{
  "redirect_url": "https://explore.pixalink.io/l/my-cafe/login/123?expires=1746519000&signature=
}
```

Redirect the customer's browser to `redirect_url`. Valid for **10 minutes**. Single-use — do not cache.

Signing the Request

Sign the **exact raw JSON bytes** you send. Do not re-serialize, pretty-print, or sort keys — the server validates against the raw body.

PHP

```php
$body = json_encode([
    'customer_phone' => '+60123456789',
    'space_id'       => 42,
    'timestamp'      => time(),
    'redirect_to'    => 'rewards', // optional — defaults to 'home'
]);

$signature = hash_hmac('sha256', $body, $clientSecret);

$response = Http::withHeaders([
    'X-Client-Id'  => $clientId,
    'X-Signature'  => $signature,
    'Content-Type' => 'application/json',
])->send('POST', 'https://explore.pixalink.io/api/sso/redirect', [
    'body' => $body,
]);

return redirect($response->json('redirect_url'));
```

Node.js

```
const crypto = require('crypto');

const body = JSON.stringify({
  customer_phone: '+60123456789',
  space_id: 42,
  timestamp: Math.floor(Date.now() / 1000),
  redirect_to: 'rewards', // optional — defaults to 'home'
});

const signature = crypto
  .createHmac('sha256', clientSecret)
  .update(body)
  .digest('hex');

const response = await fetch('https://explore.pixalink.io/api/sso/redirect', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'X-Client-Id':  clientId,
    'X-Signature':  signature,
  },
  body,
});

const { redirect_url } = await response.json();
res.redirect(redirect_url);
```

New Customer Handling

If no customer exists for the given phone number in your organisation, Pixalink **creates one automatically**. The new record has no name or date of birth.

When the customer follows the signed URL, the portal checks profile completeness:

- **Incomplete profile** (missing name or date of birth) → customer sees a registration form before the portal home
- **Complete profile** → customer lands directly on the portal home

First-time SSO logins always require a brief profile setup. All subsequent logins are seamless.

Error Reference

Status	Cause	Fix
`401`	Missing `X-Client-Id` or `X-Signature`	Add both headers
`401`	Invalid or wrong signature	Verify you're signing the raw body with the correct secret
`401`	Timestamp outside ±60s window	Sync your server clock via NTP
`401`	Client is revoked	Contact Pixalink support to re-activate
`403`	Client type is not `sso`	Your credential is a standard OAuth2 client — request an SSO client
`403`	OAuth2 not enabled for organisation	Contact Pixalink support
`404`	Passport feature globally disabled	Contact Pixalink support
`422`	Validation error	Check `customer_phone` (must be E.164), `space_id` (must belong to your org), and `redirect_to` (must be a value from the allowed list)
`429`	Rate limit exceeded	Back off — max 30 requests/minute per client

---

Good to Know

- **Clock sync is critical.** Timestamp validation rejects requests more than 60 seconds old or future-dated. Run NTP on your server. A drifted clock causes silent `401` failures that are hard to debug.
- **Phone normalisation.** Use a library like `libphonenumber` (available for PHP, JS, Python, Swift, Kotlin) to normalise numbers to E.164 before sending. `0123456789` will fail validation — `+60123456789` will not.
- **One `space_id` per call.** If your organisation has multiple outlets, pass the space the customer is associated with. For single-outlet setups, always use the same ID. Omit `space_id` entirely if your organisation uses the organisation-wide portal — Pixalink will use your configured default outlet automatically.
- **Redirect URL is single-use.** Once the customer's browser follows the URL and the portal session is created, the signed URL cannot be used again. Do not cache or share it.

Code Examples

Kotlin (Android)

```kotlin
import javax.crypto.Mac
import javax.crypto.spec.SecretKeySpec
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import org.json.JSONObject

fun ssoRedirect(
    clientId: String,
    clientSecret: String,
    customerPhone: String,
    spaceId: Int? = null,
    redirectTo: String = "home",
) {
    val payload = JSONObject().apply {
        put("customer_phone", customerPhone)
        if (spaceId != null) put("space_id", spaceId)
        put("timestamp", System.currentTimeMillis() / 1000L)
        put("redirect_to", redirectTo)
    }

    val body = payload.toString()
    val signature = hmacSha256(body, clientSecret)

    val request = Request.Builder()
        .url("https://explore.pixalink.io/api/sso/redirect")
        .addHeader("X-Client-Id", clientId)
        .addHeader("X-Signature", signature)
        .addHeader("Content-Type", "application/json")
        .post(body.toRequestBody("application/json".toMediaType()))
        .build()

    val response = OkHttpClient().newCall(request).execute()
    val redirectUrl = JSONObject(response.body!!.string()).getString("redirect_url")
    // Open redirectUrl in the device browser or a WebView
}

fun hmacSha256(data: String, secret: String): String {
    val mac = Mac.getInstance("HmacSHA256")
    mac.init(SecretKeySpec(secret.toByteArray(), "HmacSHA256"))
    return mac.doFinal(data.toByteArray()).joinToString("") { "%02x".format(it) }
}
```

Required dependency — add to `build.gradle`:

```
implementation("com.squareup.okhttp3:okhttp:4.12.0")
```

Swift (iOS)

```swift
import CommonCrypto
import Foundation

func ssoRedirect(
    clientId: String,
    clientSecret: String,
    customerPhone: String,
    spaceId: Int? = nil,
    redirectTo: String = "home"
) async throws -> URL {
    var payload: [String: Any] = [
        "customer_phone": customerPhone,
        "timestamp": Int(Date().timeIntervalSince1970),
        "redirect_to": redirectTo,
    ]
    if let spaceId { payload["space_id"] = spaceId }

    let body = try JSONSerialization.data(withJSONObject: payload)
    let bodyString = String(data: body, encoding: .utf8)!
    let signature = hmacSHA256(message: bodyString, secret: clientSecret)

    var request = URLRequest(url: URL(string: "https://explore.pixalink.io/api/sso/redirect")!)
    request.httpMethod = "POST"
    request.httpBody = body
    request.setValue(clientId, forHTTPHeaderField: "X-Client-Id")
    request.setValue(signature, forHTTPHeaderField: "X-Signature")
    request.setValue("application/json", forHTTPHeaderField: "Content-Type")

    let (data, _) = try await URLSession.shared.data(for: request)
    let json = try JSONSerialization.jsonObject(with: data) as! [String: Any]
    return URL(string: json["redirect_url"] as! String)!
    // Open the returned URL with UIApplication.shared.open() or ASWebAuthenticationSession
}

func hmacSHA256(message: String, secret: String) -> String {
    let keyBytes = Array(secret.utf8)
    let msgBytes = Array(message.utf8)
    var digest = [UInt8](repeating: 0, count: Int(CC_SHA256_DIGEST_LENGTH))
    CCHmac(CCHmacAlgorithm(kCCHmacAlgSHA256), keyBytes, keyBytes.count, msgBytes, msgBytes.count
    return digest.map { String(format: "%02x", $0) }.joined()
}
```

`CommonCrypto` is bundled with iOS — no extra dependency needed.

Flutter (Dart)

```dart
import 'dart:convert';
import 'package:crypto/crypto.dart';
import 'package:http/http.dart' as http;

Future<Uri> ssoRedirect({
  required String clientId,
  required String clientSecret,
  required String customerPhone,
  int? spaceId,
  String redirectTo = 'home',
}) async {
  final payload = <String, dynamic>{
    'customer_phone': customerPhone,
    'timestamp': DateTime.now().millisecondsSinceEpoch ~/ 1000,
    'redirect_to': redirectTo,
  };
  if (spaceId != null) payload['space_id'] = spaceId;

  final body = jsonEncode(payload);
  final signature = Hmac(sha256, utf8.encode(clientSecret))
      .convert(utf8.encode(body))
      .toString();

  final response = await http.post(
    Uri.parse('https://explore.pixalink.io/api/sso/redirect'),
    headers: {
      'X-Client-Id': clientId,
      'X-Signature': signature,
      'Content-Type': 'application/json',
    },
    body: body,
  );

  final json = jsonDecode(response.body) as Map<String, dynamic>;
  return Uri.parse(json['redirect_url'] as String);
  // Launch with url_launcher: launchUrl(redirectUri)
}
```

Required dependencies — add to `pubspec.yaml`:

```yaml
dependencies:
  http: ^1.2.0
  crypto: ^3.0.0
```