OwnRewards

LOYALTY INTEGRATION API

Bring OwnRewards points and coupons into your checkout

Let your website, app or billing software recognise members, show their points, apply rewards as discounts and record every sale — automatically and securely.

iEYAL

Version 1.1 · 19 November 2026 · Published by iEYAL Solutions

Confidential — for evaluation. Shared with prospective customers and partners to evaluate an OwnRewards integration. Please do not publish or forward. This document contains no keys or passwords: each merchant creates their own in the OwnRewards admin panel.

AT A GLANCE

What the OwnRewards API lets you do

The OwnRewards API is a secure connection that lets an online store, app or billing system run the merchant's loyalty programme at checkout — so shoppers earn and spend points everywhere, not only at the counter.

👤 Recognise members

Find a shopper by mobile number or email, or enrol them in the loyalty programme in one step.

⭐ Show points and tier

Display the member's points balance, tier and available coupons and rewards right in your checkout.

🎯 Show "you'll earn" points

Tell the shopper how many points this order will earn — a proven nudge to complete the purchase.

🏷️ Apply points, coupons and rewards

Turn points, coupon codes or rewards into a discount, following the merchant's own rules.

🧾 Record the sale safely

After payment, record the order so points are earned and spent in one step — never twice.

↩️ Handle returns

If an order is cancelled or returned, points are reversed automatically.

Who uses it

If you are…	You can…
An e-commerce website	Add "use your points" and "you'll earn N points" to your cart and checkout.
A mobile app	Show the member's card, balance and rewards inside your app.
Billing or ordering software	Record each sale against the member and apply their rewards as a discount.
A marketing or CRM tool	Issue coupons to members and get notified about sign-ups, sales, birthdays and win-backs.

Getting started — 3 steps

1

Create a key

The merchant opens Connect → API & Webhooks, picks the outlet and the "Storefront" permission preset.

2

Share it securely

The merchant gives the key and secret to the integration partner directly — never by public chat.

3

Connect and test

The partner's developer follows the technical section of this guide. We provide a test account on request.

SECURITY

How the connection is kept safe

The merchant controls who connects, what each connection may do, and can close it at any time.

1

The merchant creates the key

Keys are made in the OwnRewards admin panel (**Connect → API & Webhooks**). leyal Solutions never sends keys by email or in documents.

2

A key and a secret, like a username and password

Both are required on every request. The secret is shown only once and is never stored in readable form.

3

Only the permissions you give

Each key is limited to chosen actions — for example "read members" and "record sales" — and nothing else.

4

Locked to one business and outlet

A key works only for the merchant that created it, and can be locked to a single outlet.

5

No double counting

Every recorded sale carries a unique reference, so a repeated request can never earn or spend points twice.

6

Revoke any time, signed updates

Revoke a key and it stops at once. Notifications we send you are digitally signed so you know they came from OwnRewards. All traffic is HTTPS and rate-limited.

Every feature in this document requires a key. API keys are included in the **Pro** (up to 10 keys) and **Enterprise** (unlimited) plans. Notifications (webhooks) are available from the **Growth** plan.

HOW IT WORKS

A loyalty checkout, start to finish

1. Recognise
Find or enrol the shopper by mobile number.

→

2. Show
Points balance, tier and rewards.

→

3. Apply
Use points, a coupon or a reward; see the new total.

→

4. Record
After payment, record the sale — points earned and spent.

→

5. Return
Cancelled? Points are reversed.

OwnRewards applies the merchant's own earning, tier, coupon and stacking rules, so online sales behave exactly like in-store sales.

CAPABILITIES

Everything you can do, in plain words

Each feature below shows what it does, when you would use it, and — for developers — the exact address and permission it needs.

MEMBERS

Find or enrol a member

Looks up the shopper by mobile number, email or your own customer id. If they are not a member yet, they are enrolled automatically (this can be switched off).

Use it for: sign-in, checkout, "join our rewards" buttons.

Developers: `POST /v1/loyalty/identify` · permission `customer:write`

MEMBERS

Show points balance and tier

Returns the member's current points, points earned and spent so far, tier (e.g. Silver, Gold) and progress to the next tier.

Use it for: a loyalty card in your app, "you have 1,250 points" in checkout.

Developers: `GET /v1/loyalty/customers/{customerRef}/summary` · permission `customer:read`

MEMBERS

Show available coupons and rewards

Lists the coupons and rewards the member can use right now.

Use it for: a "your rewards" list the shopper can pick from at checkout.

Developers: `GET /v1/loyalty/customers/{customerRef}/rewards` · permission `customer:read`

CHECKOUT

Show how many points an order will earn

Calculates the points this cart would earn, including tier bonuses and product multipliers. Nothing is saved.

Use it for: "Complete this order and earn 63 points".

Developers: `POST /v1/loyalty/earn/preview` · permission `customer:read`

CHECKOUT

Apply points, a coupon or a reward and get the new total

Works out the discount and the final amount to charge, following the merchant's rules on which offers can be combined. Explains anything that couldn't be applied. Nothing is spent yet.

Use it for: "Use 200 points — save ₹20" before payment.

Developers: `POST /v1/loyalty/cart/quote` · permission `customer:read`

CHECKOUT

Record the sale (points earned and spent)

After the customer has paid, records the order: points are earned, and any points, coupon or reward used are redeemed — all in one safe step that can't happen twice.

Use it for: the moment payment succeeds.

Developers: `POST /v1/loyalty/cart/commit` (or `/cart/commit-by-phone`) · permission `bill:write`

COUPONS

Check a coupon code

Tells you whether a code is valid for this member and cart, and how much it takes off — or why it can't be used.

Use it for: the "Apply coupon" box.

Developers: `POST /v1/loyalty/coupons/validate` · permission `coupon:read`

COUPONS

Give a coupon to a shopper

Issues one of the merchant's coupons to a shopper by mobile number or email, enrolling them if needed.

Use it for: welcome offers, survey thank-yous, campaign rewards.

Developers: `POST /v1/loyalty/coupons/issue` · permission `coupon:write`

RETURNS

Reverse points for a cancelled or returned order

Takes back points earned on the order and gives back points that were spent.

Use it for: cancellations and returns in your order system.

Developers: `POST /v1/loyalty/order/return` · permission `bill:write`

UPDATES

Instant notifications (webhooks)

OwnRewards calls your system automatically when something happens. The merchant switches these on under **Connect → API & Webhooks**. Every notification is signed and is retried if your system is temporarily down.

Notification	Sent when
Member joined / updated	A shopper joins or their details change
Sale recorded	A bill or order is recorded
Sale reversed	A bill is refunded or an order cancelled
Visit milestones	First visit, every Nth visit, milestone reached
Lapsed and won back	A member goes inactive, or returns
Special dates	Birthdays and anniversaries (on the day and coming up)

Use it for: CRM sync, personalised messages, marketing automation.

FOR DEVELOPERS

Technical reference

This section is for the partner's developer. Customer success teams can skip it.

Basics

Item	Value
Base address	`https://api.ownrewards.ai` — all calls start with `/v1/loyalty`
Format	JSON over HTTPS, body up to 100 KB
Outlet	Send one of `restId` / `eateryBranchCode` (your branch code) or `outletId`; if none, the key's own outlet is used
Limits	120 requests per minute → `429 rate_limited` with `Retry-After`

HEADERS ON EVERY REQUEST

```
X-API-Key: ownrewards_<key>
X-API-Secret: <secret>
Content-Type: application/json
```

SUCCESS RESPONSE

```
{
  "status": "success",
  "data": {}
}
```

ERROR RESPONSE

```
{
  "status": "failed",
  "message": "Not enough points",
  "code": "insufficient_points",
  "details": {
    "requested": 500,
    "available": 320
  }
}
```

Cart object — used by earn preview, quote, coupon check and commit (shown as `{ …cart object… }` below):

CART OBJECT

```
{
  "orderId": "100234",
  "subtotal": 1200,
  "total": 1260,
  "taxTotal": 60,
  "items": [
    {
      "itemId": "SKU-1",
      "itemName": "Mysore Pak",
      "category": "Sweets",
      "quantity": 2,
      "pricePerUnit": 300,
      "totalPrice": 600
    }
  ]
}
```

Applied object — what the shopper chose to use (all optional):

APPLIED OBJECT

```
{
  "points": 200,
  "couponCode": "WELCOME10",
  "rewardId": "<customerRewardId>"
}
```

Members

POST `/v1/loyalty/identify`

Find or enrol a member by phone, email or your own customer id.

REQUEST BODY

```
{
  "phone": "9876543210",
  "countryCode": "91",
  "name": "Priya",
  "createIfMissing": true
}
```

RESPONSE 200

```
{
  "status": "success",
  "data": {
    "customerRef": "<customerRef>",
    "customerId": "<id>",
    "customer": {
      "name": "Priya",
      "phone": "919876543210",
      "tier": "Silver"
    }
  }
}
```

GET `/v1/loyalty/customers/{customerRef}/summary`

Points balance and tier.

RESPONSE 200

```
{
  "status": "success",
  "data": {
    "points": {
      "balance": 1250,
      "earned": 1800,
      "redeemed": 500,
      "expired": 50
    },
    "tier": {
      "current": "Silver",
      "next": "Gold"
    },
    "couponsCount": 2,
    "rewardsCount": 1
  }
}
```

`GET` `/v1/loyalty/customers/{customerRef}/rewards`

Coupons and rewards the member can use now.

RESPONSE `200`

```
{
  "status": "success",
  "data": {
    "coupons": [
      {
        "name": "Welcome offer",
        "code": "WELCOME10-7KQ2",
        "expiresAt": "2026-12-31"
      }
    ],
    "rewards": [
      {
        "_id": "<customerRewardId>",
        "name": "Free Badam Milk",
        "points": 300
      }
    ]
  }
}
```

Send a reward's `_id` as `applied.rewardId`.

Checkout

`POST` `/v1/loyalty/earn/preview`

Points this cart would earn. Nothing is saved.

REQUEST BODY

```
{
  "customerRef": "<customerRef>",
  "cart": { …cart object… }
}
```

RESPONSE `200`

```
{
  "status": "success",
  "data": {
    "pointsToEarn": 63,
    "basePoints": 60,
    "bonusPoints": 3,
    "multipliers": {
      "product": 1,
      "tier": 1.05,
      "combined": 1.05
    }
  }
}
```

POST /v1/loyalty/cart/quote

Discount and final amount with points, coupon or reward applied. Nothing is spent yet.

REQUEST BODY

```
{
  "customerRef": "<customerRef>",
  "cart": { …cart object… },
  "applied": {
    "points": 200
  }
}
```

RESPONSE 200

```
{
  "status": "success",
  "data": {
    "applied": [
      {
        "promotionType": "points",
        "label": "200 points",
        "discountAmount": 20
      }
    ],
    "rejected": [],
    "discount": {
      "total": 20,
      "finalAmount": 1240
    },
    "wallet": {
      "balancePoints": 1250,
      "redeemablePoints": 1000
    },
    "earn": {
      "pointsToEarn": 62
    }
  }
}
```

POST /v1/loyalty/cart/commit

Record the paid sale — points earned and spent in one step. Call once, after payment succeeds.

REQUEST HEADERS

```
Idempotency-Key: order-100234
```

REQUEST BODY

```
{
  "customerRef": "<customerRef>",
  "externalOrderId": "100234",
  "cart": { …cart object… },
  "applied": {
    "points": 200
  }
}
```

RESPONSE 201


```
{
  "status": "success",
  "data": {
    "status": "completed",
    "orderId": "100234",
    "billId": "<bill id>",
    "pointsEarned": 62,
    "pointsRedeemed": 200,
    "balancePoints": 1112,
    "committedAt": "2026-11-18T10:15:00Z"
  }
}
```


`Idempotency-Key` is required — one unique value per order. Retrying with the same key returns the first result (`"replay": true`) and never double-counts. `/v1/loyalty/cart/commit-by-phone` takes `customerPhoneNo` + `countryCode` instead of `customerRef` for existing members.

Coupons and returns

POST `/v1/loyalty/coupons/validate`

Is this code valid for this member and cart, and how much does it take off?

REQUEST BODY

```
{
  "customerRef": "<customerRef>",
  "code": "WELCOME10",
  "cart": { …cart object… }
}
```

RESPONSE 200


```
{
  "status": "success",
  "data": {
    "valid": true,
    "code": "WELCOME10",
    "discount": {
      "type": "percentage",
      "value": 10,
      "discountAmount": 120,
      "label": "10% off"
    }
  }
}
```


RESPONSE — NOT VALID


```
{
  "status": "success",
  "data": {
    "valid": false,
    "code": "WELCOME10",
    "reason": "Minimum order value is ₹500"
  }
}
```

POST /v1/loyalty/coupons/issue

Give a coupon to a shopper (enrols them if needed).

REQUEST BODY

```
{
  "code": "WELCOME10",
  "phone": "9876543210",
  "countryCode": "91",
  "name": "Priya"
}
```

RESPONSE 201

```
{
  "status": "success",
  "data": {
    "memberCreated": false,
    "customerRef": "<customerRef>",
    "coupon": {
      "name": "Welcome offer",
      "code": "WELCOME10-7KQ2",
      "status": "issued",
      "expiresAt": "2026-12-31"
    }
  }
}
```

POST /v1/loyalty/order/return

Reverse points for a cancelled or returned order. Use the same order id as at commit.

REQUEST BODY

```
{
  "orderId": "100234",
  "mobile": "9876543210",
  "purchase_date": "2026-11-18",
  "reason": "Customer cancelled"
}
```

Returns `{ "status": "success", "data": { …refund result… } }`.

Webhooks

OwnRewards sends a signed `POST` to your https address. Reply 2xx within 15 seconds; failures are retried up to 8 times (2 minutes → 1 hour). Use `eventId` to ignore duplicates.

HEADERS

```
X-OwnRewards-Event: bill_created
X-OwnRewards-Timestamp: 1763460900000
X-OwnRewards-Signature: <hex HMAC-SHA256>
```

BODY

```
{
  "event": "bill_created",
  "timestamp": "1763460900000",
  "data": {
    "eventId": "<event id>",
    "customerId": "<customer id>",
    "outletId": "<outlet id>"
  }
}
```

Verify: HMAC-SHA256 with your `whsec_…` secret over `timestamp + "." + rawBody`, compared with `X-OwnRewards-Signature`; reject timestamps older than 5 minutes.

NODE.JS — VERIFY A WEBHOOK

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

function verify(rawBody, headers, secret) {
  const ts = headers['x-ownrewards-timestamp'];
  const expected = crypto.createHmac('sha256', secret)
    .update(`${ts}.${rawBody}`).digest('hex');
  const given = headers['x-ownrewards-signature'] || '';
  return given.length === expected.length
    && crypto.timingSafeEqual(Buffer.from(given), Buffer.from(expected))
    && Math.abs(Date.now() - Number(ts)) < 5 * 60 * 1000;
}
```

Error codes

HTTP	code	Meaning
400	`validation_error`	Missing or wrong field (`details` lists them)
400	`idempotency_key_required`	Commit sent without `Idempotency-Key`
400	`insufficient_points`	Not enough points to redeem
400	`reward_unavailable`, `coupon_invalid`, `discount_exceeds_bill`	The discount can't be applied
401	`unauthorized`	Key or secret missing, wrong or revoked
403	`forbidden`, `outlet_scope_mismatch`	Missing permission, or key locked to another outlet
404	`customer_not_found`, `outlet_not_found`, `coupon_not_found`	Not found
409	`duplicate_order`, `coupon_already_issued`	Already done
429	`rate_limited`	Slow down — see `Retry-After`
500	`internal_error`	Temporary problem — retry (commit is safe to retry with the same key)

Confidential — for evaluation only. Please do not publish or forward this document.
Next steps. Ask your iEYAL Solutions account manager for a test account. Questions during integration are answered by our technical team.