

OAuth2

Author Kodmyran AB

OAuth2 is available from Kodmyran Commerce version 7.13. Earlier versions accept API keys only.

Introduction

OAuth2 lets an application act on behalf of a user of Kodmyran Commerce without ever seeing their password or API key. The user logs in, sees what the application asks for, and approves it. The application then receives a token that works for that user, limited to what was approved, and the user can withdraw it at any time.

Use OAuth2 when you build an application that other merchants connect to their accounts: a dashboard, an app in a marketplace, an AI assistant. For an integration the merchant runs themselves, such as their own ERP connection, an API key is still the simpler choice.

In short:

- The only flow is **Authorization Code with PKCE** (RFC 6749, RFC 7636), with S256 as the challenge method.
- All clients are **public**: they have a client ID and no secret. There is no client credentials, implicit or password grant, and no OpenID Connect.
- A token can never do more than the user who approved it. **What it can do is the user's role intersected with the scopes approved.**
- OAuth2 answers only on the account domain, `https://<account>.kodmyran.io`, never on the shop's own domain.
- The entity, media, service and report APIs accept tokens. The integration and headless APIs do not.

Before you start

Registering your application

Your application needs a client ID in each account it connects to. There are two ways to get one:

- **An administrator creates it** in the admin under *Configuration* → *User* → *OAuth2 applications* (*OAuth2-applikationer*), giving your application's name and redirect URIs. The client ID is shown once saved.
- **Your application registers itself** at the registration endpoint (RFC 7591), if an administrator has switched on dynamic registration under *OAuth2 settings* (*OAuth2-inställningar*). It is off by default. Applications that registered themselves are marked as unverified on the consent page.

A self-registration request names the application and its redirect URIs:

POST `https://testaccount.kodmyran.io/admin/oauth2/register`
Content-Type: `application/json`

```
{
  "client_name": "My Dashboard",
  "redirect_uris": ["https://dashboard.example.com/oauth/callback"],
  "grant_types": ["authorization_code", "refresh_token"],
  "response_types": ["code"],
  "token_endpoint_auth_method": "none"
}
```

The response contains your `client_id`. Registration is limited to 10 per address and hour.

Redirect URIs

- A redirect URI must match exactly. There is no prefix or pattern matching.
- It must use https. Plain http is accepted only to a loopback address (localhost, 127.0.0.1 or `:::1`), for local tools.
- It may not contain a fragment, and is at most 500 characters long.

Who can approve

The user approving your application needs two things in Kodmyran Commerce:

- **API key delegation** (*API nyckel delegering*) on their admin user. Approving an application is the same act as creating an API key that works as themselves, so it needs the same right.

- A role holding **Remote call: Read**, like any other API caller.

A user without delegation sees a page saying they cannot connect applications.

The flow

1. Send the user to the authorization endpoint

Create a random code verifier, derive the code challenge from it (S256), and send the user's browser to:

```
https://testaccount.kodmyran.io/admin/oauth2/authorize
  ?response_type=code
  &client_id=<your client id>
  &redirect_uri=https://dashboard.example.com/oauth/callback
  &scope=orders:report fulfilment:report
  &state=<random value you check later>
  &code_challenge=<challenge>
  &code_challenge_method=S256
```

The user logs in if needed, and sees a consent page naming your application and listing what it asks for. Consent is asked every time; nothing is remembered between authorizations.

If the client ID or redirect URI is wrong, the error is shown to the user and nothing is sent to the redirect URI. Any other error in the request is sent back to the redirect URI as error and error_description, as the standard requires, e.g. invalid_scope for an unknown scope.

2. Receive the code

When the user approves, the browser returns to your redirect URI with code, state and iss (RFC 9207). Check that state is the value you sent. If the user denies, you receive error=access_denied instead.

3. Exchange the code for tokens

```
POST https://testaccount.kodmyran.io/admin/oauth2/token
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=authorization_code&code=<code>&redirect_uri=https://dashboard.example.com/oauth/c
```

Response:

```
{
  "access_token": "...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "...",
  "scope": "orders:report fulfilment:report"
}
```

The client ID may be sent in the body as above, or as HTTP Basic with an empty password.

4. Call the API

Send the access token in the Authorization header:

```
GET https://testaccount.kodmyran.io/admin/api/entity/order
Authorization: Bearer <access token>
```

The token is checked on every call, so it stops working at once if the user withdraws it, is deactivated or loses API key delegation, or if an administrator disables your application.

5. Refresh

An access token lives for an hour. Get a new pair with the refresh token:

```
POST https://testaccount.kodmyran.io/admin/oauth2/token
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=refresh_token&refresh_token=<refresh token>&client_id=<your client id>
```

A refresh may ask for fewer scopes than were approved by adding scope, never more.

A refresh token can be used once. Every refresh returns a new refresh token, and you must store it before using the new access token. Presenting a refresh token that has already been used is treated as theft, and withdraws the user's whole approval: the user has to approve your application again.

What	Lifetime	Use
Authorization code	5 minutes	Once
Access token	1 hour	Any number of calls
Refresh token	30 days	Once; each refresh returns a new one

An application that refreshes at least every 30 days keeps working without the user, e.g. for a nightly job.

6. Revoke

When the user disconnects your application, revoke the refresh token. That ends everything issued under the approval:

```
POST https://testaccount.kodmyran.io/admin/oauth2/revoke
Content-Type: application/x-www-form-urlencoded
```

```
token=<refresh token>&client_id=<your client id>
```

The answer is 200 whether or not the token was known.

Users can also see and withdraw their approvals in the admin, under *Connected applications* (*Kopplade applikationer*).

Binding a token to the API

You may bind a token to the REST API by adding resource to the authorization and token requests (RFC 8707):

```
resource=https://testaccount.kodmyran.io/admin/api/
```

A bound token is refused by any other service that checks its audience, so it cannot be replayed elsewhere. A token without a binding is accepted by the API too.

Scopes

A scope is written as *domain:verb*, e.g. *orders:read*. Ask for several by separating them with spaces, as the standard requires: **products:read products:update orders:read**. There is no shorthand for several verbs at once, and no scope that grants everything.

Domains

The data of Kodmyran Commerce is divided into domains:

Domain	Covers
products	Products, product groups and option lists, including prices and supplier rows edited on the product
categories	The category tree
brands	Brands and manufacturers
reviews	Product reviews
pricing	Price lists, agreements, discounts and fees
inventory	Warehouses, stock and stock movements
orders	Orders, quotations, returns, subscriptions and payments
fulfilment	Picking, shipping and delivery
purchasing	Purchase orders, purchase invoices and suppliers
customers	Customers, customer groups and their lists
marketing	Campaigns, deals, bonus levels and recommendations
content	Pages, blogs, views and search engine settings
finance	Invoices, ledgers, VAT and fiscal years
settings	Store settings such as attributes, countries, currencies, languages and units
integrations	Webhooks and sync views
media	Uploaded images and documents

An account may define domains of its own. The authoritative list for an account is its discovery document, see [Discovery](#).

Verbs

Verb	Allows	Used by
create	Creating objects (POST)	Entity and media API

Verb	Allows	Used by
read	Reading objects (GET), and reports that hand out records	Entity, media and report API
update	Changing objects (PUT)	Entity and media API
delete	Deleting objects (DELETE)	Entity and media API
call	Running services	Service API
report	Running reports that give figures	Report API

Entity API

Each entity belongs to one domain, and the verb follows the HTTP method of the call. A sub-entity follows its parent: products:update also covers the prices and supplier rows edited on a product. Through a price list, the same rows need pricing:update.

Some entities can never be reached with a token, such as admin users, roles, API keys and anything holding a secret. Reading GET /admin/api/entity/ with a token lists only the entities a token can reach.

Media API

The media API uses the media domain:

Scope	Call
media:read	GET, the details of a file
media:create	PUT or POST to a path where no file exists
media:update	PUT or POST that replaces an existing file
media:delete	DELETE

The user's role also needs the *Remote call: Upload* permission, as for an API key.

Service API

Services are run under *domain:call*, one verb whatever the HTTP method. The role must also have been granted the service. A service without a domain cannot be run with a token.

Scope	Services
fulfilment:call	DeliverOrder, ReturnOrder, BrokerPrint, Transfer2Broker, FetchParcels, ShipmentBatch
inventory:call	ConfirmPickOrder, DeliverPurchaseOrder, StockTaking
purchasing:call	PostPurchaseOrder
orders:call	FlagOrder
customers:call	GetTicketing

The ReturnOrder service receives returned goods into the warehouse, the reverse of DeliverOrder, and is therefore fulfilment. A return request registered as a ReturnOrder object through the entity API belongs to orders.

Report API

Reports that give figures are run under *domain:report*. They give totals and counts, never individual records, so a dashboard can show sales without being able to read a single order. A report that hands out records, such as Top-Spenders with its customer names, needs the domain's read scope instead. The role must also have been granted the report.

Scope	Reports
orders:report	SalesByValue, AverageOrderValue, PaymentMethods, FreightMethods, FreightValue, PlacedOrders, ReturningCustomer
fulfilment:report	WaitingOrders, PrintedOrders, ReplacementsToday, DeliveredOrders, ProductReturns, TopPickers, TopPackers, PickTime, BatchPicklists
products:report	ProductStatistics, ProductOutOfStock, UnsoldProducts
inventory:report	Currentstockvalue
purchasing:report	ExpectedInboundOrders, PurchaseOrderBuys, PurchaseorderBySupplier
finance:report	ProvisionSales
customers:read	TopSpenders

The fulfilment reports show picking and packing per employee, and Provision-Sales shows sales per salesperson; the consent page says so. The BI streams

in the BusinessIntelligence group cannot be run with a token. Every output format is available.

Errors

Situation	Answer
Unknown, expired or withdrawn token, or one bound to another resource	401, WWW-Authenticate: Bearer error="invalid_token"
The user's role does not allow the call	403, as for an API key
The token lacks the scope	403, WWW-Authenticate: Bearer error="insufficient_scope", scope="orders:read"
An entity, service or report a token cannot reach	403
The integration API	403, "This API cannot be used with an OAuth2 token"

The scope named in an `insufficient_scope` answer is the one that call needs, so your application can ask the user for it.

At the token endpoint, errors follow RFC 6749: `invalid_grant` for a code or refresh token that is wrong, expired or already used, `invalid_client` for an unknown or disabled client, `invalid_scope` and `unsupported_grant_type`.

Repeated calls with an invalid token lock out your address for a while, the same way repeated calls with an invalid API key do.

Discovery

Each account describes its OAuth2 server and the scopes it supports in two standard documents:

<https://testaccount.kodmyran.io/.well-known/oauth-authorization-server>
<https://testaccount.kodmyran.io/.well-known/oauth-protected-resource/admin/api/>

The first (RFC 8414) lists the endpoints, the supported grant types and challenge methods, and every scope that can be asked for in that account. The second (RFC 9728) describes the REST API as a protected resource. Libraries that support discovery need only the account domain.

Endpoint	Purpose
/admin/oauth2/authorize	The user logs in and approves
/admin/oauth2/token	Exchange a code, or refresh, for tokens
/admin/oauth2/revoke	Revoke a token (RFC 7009)
/admin/oauth2/register	Register an application (RFC 7591), when switched on