



- [Getting Started](#)
- [API Reference](#)
- [Billing Requests](#)
- [Create a sandbox account](#)
  
- [Overview](#)
  - [Supported Direct Debit Schemes](#)
  - [Getting Started](#)
  - [Restrictions](#)
    - [Payment page restrictions](#)
    - [Creditor management restrictions](#)
  - [Anatomy](#)
    - [Mandate setup](#)
    - [Payment creation](#)
    - [How timings work](#)
  - [Backwards Compatibility](#)
  - [Changelog](#)
  - [Client Libraries](#)
  
- [API Usage](#)
  - [Making Requests](#)
    - [Base URLs](#)
    - [Authentication](#)
    - [Versions](#)
    - [MIME Types](#)
    - [PUT, PATCH and DELETE](#)
    - [Rate Limiting](#)
    - [Timeouts](#)
    - [Idempotency Keys](#)
    - [Optional properties](#)
  - [Dates and Times](#)
  - [Cursor Pagination](#)
    - [Options](#)
    - [Response](#)
  - [Response Codes](#)
  - [Scenario Simulators](#)
  - [Errors](#)
    - [Error types](#)
    - [GoCardless errors](#)
    - [Invalid API usage errors](#)
    - [Invalid state errors](#)
    - [Validation errors](#)
  
- [Billing Requests](#)
  - [Bank Authorisations](#)
    - [Create a Bank Authorisation](#)
    - [Get a Bank Authorisation](#)
  - [Billing Requests](#)
    - [Create a Billing Request](#)
    - [Collect customer details](#)
    - [Collect bank account details](#)
    - [Confirm the payer details](#)
    - [Fulfil a Billing Request](#)
    - [Cancel a Billing Request](#)
    - [List Billing Requests](#)
    - [Get a single Billing Request](#)
    - [Notify the customer](#)
    - [Trigger fallback](#)
    - [Change currency](#)
    - [Select institution for a Billing Request](#)
  - [Billing Request Flows](#)
    - [Create a Billing Request Flow](#)
    - [Initialise a Billing Request Flow](#)
  - [Billing Request Templates](#)
    - [List Billing Request Templates](#)
    - [Get a single Billing Request Template](#)
    - [Create a Billing Request Template](#)
    - [Update a Billing Request Template](#)
  - [Billing Request with Actions](#)
    - [Create a Billing Request with Actions](#)
  - [Institutions](#)
    - [List Institutions](#)
    - [List institutions](#)
  
- [Core Endpoints](#)
  - [Balances](#)
    - [List balances](#)
  - [Bank Account Details](#)
    - [Get encrypted](#)
  - [Blocks](#)
    - [Create a block](#)
    - [Get a single block](#)
    - [List multiple blocks](#)
    - [Disable a block](#)
    - [Enable a block](#)
    - [Create blocks batch](#)
  - [Creditors](#)

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Accepter](#)
[Gérer](#)

[Consultez notre politique de confidentialité](#)


- [Create a creditor](#)
- [List creditors](#)
- [Get a single creditor](#)
- [Update a creditor](#)
- [Creditor Bank Accounts](#)
  - [Create a creditor bank account](#)
  - [List creditor bank accounts](#)
  - [Get a single creditor bank account](#)
  - [Disable a creditor bank account](#)
  - [Validate Creditor Bank Account](#)
- [Currency Exchange Rates](#)
  - [List exchange rates](#)
- [Customers](#)
  - [Create a customer](#)
  - [List customers](#)
  - [Get a single customer](#)
  - [Update a customer](#)
  - [Remove a customer](#)
- [Customer Bank Accounts](#)
  - [Create a customer bank account](#)
  - [List customer bank accounts](#)
  - [Get a single customer bank account](#)
  - [Update a customer bank account](#)
  - [Disable a customer bank account](#)
- [Customer Notifications](#)
  - [Handle a notification](#)
- [Events](#)
  - [List events](#)
  - [Get a single event](#)
  - [Reconciling Payouts with Events](#)
- [Exports](#)
  - [Get a single export](#)
  - [List exports](#)
- [Instalment Schedules](#)
  - [Create \(with dates\)](#)
  - [Create \(with schedule\)](#)
  - [List instalment schedules](#)
  - [Get a single instalment schedule](#)
  - [Update an instalment schedule](#)
  - [Cancel an instalment schedule](#)
- [Logos](#)
  - [Create a logo associated with a creditor](#)
- [Mandates](#)
  - [Create a mandate](#)
  - [List mandates](#)
  - [Get a single mandate](#)
  - [Update a mandate](#)
  - [Cancel a mandate](#)
  - [Reinstate a mandate](#)
- [Mandate Imports](#)
  - [Create a new mandate import](#)
  - [Get a mandate import](#)
  - [Submit a mandate import](#)
  - [Cancel a mandate import](#)
- [Mandate Import Entries](#)
  - [Add a mandate import entry](#)
  - [List all mandate import entries](#)
- [Negative Balance Limits](#)
  - [List negative balance limits](#)
- [Outbound Payments](#)
  - [Create an outbound payment](#)
  - [Create a withdrawal outbound payment](#)
  - [Cancel an outbound payment](#)
  - [Approve an outbound payment](#)
  - [Get an outbound payment](#)
  - [List outbound payments](#)
  - [Update an outbound payment](#)
- [Payer Authorisations](#)
  - [Get a single Payer Authorisation](#)
  - [Create a Payer Authorisation](#)
  - [Update a Payer Authorisation](#)
  - [Submit a Payer Authorisation](#)
  - [Confirm a Payer Authorisation](#)
- [Payer Themes](#)
  - [Create a payer theme](#)
- [Payments](#)
  - [Create a payment](#)
  - [List payments](#)
  - [Get a single payment](#)
  - [Update a payment](#)
  - [Cancel a payment](#)
  - [Retry a payment](#)
- [Payouts](#)
  - [List payouts](#)
  - [Get a single payout](#)
  - [Update a payout](#)
- [Payout Items](#)
  - [List payout items](#)

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

 Consultez notre politique de confidentialité

- [Get all payout items in a single payout](#)
- [Redirect Flows](#)
  - [Create a redirect flow](#)
  - [Get a single redirect flow](#)
  - [Complete a redirect flow](#)
- [Refunds](#)
  - [Create a refund](#)
  - [List refunds](#)
  - [Get a single refund](#)
  - [Update a refund](#)
- [Scenario Simulators](#)
  - [Simulate a scenario](#)
- [Scheme Identifiers](#)
  - [Create a scheme identifier](#)
  - [List scheme identifiers](#)
  - [Get a single scheme identifier](#)
- [Subscriptions](#)
  - [Recurrence Rules](#)
  - [Rolling dates](#)
  - [Create a subscription](#)
  - [List subscriptions](#)
  - [Get a single subscription](#)
  - [Update a subscription](#)
  - [Pause a subscription](#)
  - [Resume a subscription](#)
  - [Cancel a subscription](#)
- [Tax Rates](#)
  - [List tax rates](#)
  - [Get a single tax rate](#)
- [Transferred Mandates](#)
  - [Get updated customer bank details](#)
- [Verification Details](#)
  - [Create a verification detail](#)
  - [List verification details](#)
- [Webhooks](#)
  - [List webhooks](#)
  - [Get a single webhook](#)
  - [Retry a webhook](#)
- [Event Actions](#)
  - [Billing Request](#)
    - [created](#)
    - [flow\\_created](#)
    - [flow\\_visited](#)
    - [flow\\_exited](#)
    - [collect\\_customer\\_details](#)
    - [select\\_institution](#)
    - [collect\\_bank\\_account](#)
    - [payer\\_details\\_confirmed](#)
    - [bank\\_authorisation\\_visited](#)
    - [bank\\_authorisation\\_authorised](#)
    - [bank\\_authorisation\\_denied](#)
    - [bank\\_authorisation\\_expired](#)
    - [bank\\_authorisation\\_failed](#)
    - [fulfilled](#)
    - [cancelled](#)
    - [choose\\_currency](#)
    - [collect\\_amount](#)
    - [payer\\_geo\\_blocked](#)
    - [failed](#)
  - [Creditor](#)
    - [updated](#)
    - [new\\_payout\\_currency\\_added](#)
    - [account\\_auto\\_frozen](#)
    - [account\\_auto\\_frozen\\_reverted](#)
    - [bounced\\_payout](#)
  - [Export](#)
    - [started](#)
    - [completed](#)
    - [failed](#)
  - [Instalment Schedule](#)
    - [created](#)
    - [creation\\_failed](#)
    - [cancelled](#)
    - [errored](#)
    - [resumed](#)
    - [completed](#)
  - [Mandate](#)
    - [created](#)
    - [customer\\_appr](#)
    - [customer\\_appr](#)
    - [active](#)
    - [cancelled](#)
    - [failed](#)
    - [transferred](#)
    - [expired](#)
    - [submitted](#)
    - [resubmission\\_requested](#)

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- [reinstated](#)
- [replaced](#)
- [consumed](#)
- [blocked](#)
- [Outbound Payment](#)
  - [created](#)
  - [cancelled](#)
  - [failed](#)
  - [set\\_to\\_pending\\_approval](#)
  - [set\\_to\\_executing](#)
  - [executed](#)
- [Payer Authorisation](#)
  - [completed](#)
  - [failed](#)
- [Payment](#)
  - [created](#)
  - [customer\\_approval\\_granted](#)
  - [customer\\_approval\\_denied](#)
  - [submitted](#)
  - [confirmed](#)
  - [chargeback\\_cancelled](#)
  - [paid\\_out](#)
  - [late\\_failure\\_settled](#)
  - [chargeback\\_settled](#)
  - [surcharge\\_fee\\_debited](#)
  - [failed](#)
  - [charged\\_back](#)
  - [cancelled](#)
  - [resubmission\\_requested](#)
  - [sds\\_settlement\\_delayed](#)
- [Payout](#)
  - [paid](#)
  - [fx\\_rate\\_confirmed](#)
  - [tax\\_exchange\\_rates\\_confirmed](#)
- [Refund](#)
  - [created](#)
  - [failed](#)
  - [paid](#)
  - [refund\\_settled](#)
  - [funds\\_returned](#)
- [Scheme Identifier](#)
  - [activated](#)
- [Subscription](#)
  - [created](#)
  - [customer\\_approval\\_granted](#)
  - [customer\\_approval\\_denied](#)
  - [amended](#)
  - [payment\\_created](#)
  - [cancelled](#)
  - [finished](#)
  - [paused](#)
  - [resumed](#)
  - [scheduled\\_pause](#)
  - [scheduled\\_pause\\_cancelled](#)
  - [scheduled\\_resume](#)
- [Success+](#)
  - [Intelligent Retries](#)
  - [Enabling Intelligent Retries](#)
    - [Individual Payments](#)
    - [Instalment Schedules](#)
    - [Subscriptions](#)
  - [Handling Failures](#)
  - [Retry Rules](#)
- [Helper Endpoints](#)
  - [Healthcheck](#)
  - [Bank Details Lookups](#)
    - [Perform a bank details lookup](#)
  - [Mandate PDFs](#)
    - [Create a mandate PDF](#)
- [Appendix](#)
  - [Webhooks](#)
    - [Overview](#)
    - [Signing Webhooks](#)
    - [Examples](#)
  - [JavaScript Flow](#)
    - [Customer Bank](#)
    - [Create a customer bank](#)
    - [Intercepting flows](#)
    - [Get a single customer bank](#)
  - [OAuth](#)
    - [The OAuth flow](#)
    - [Creating an OAuth application](#)
    - [Building an OAuth endpoint](#)
    - [Handling the response](#)
    - [Exchanging an access token](#)
    - [Making requests](#)
    - [Receiving webhooks](#)

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

- [App Fees](#)
- [Looking up an access token](#)
- [Disconnecting a user from your app](#)
- [Compliance Requirements](#)
- [Local Bank Details](#)
  - [IBAN](#)
  - [UK](#)
  - [Australia](#)
  - [Austria](#)
  - [Belgium](#)
  - [Canada](#)
  - [Cyprus](#)
  - [Denmark](#)
  - [Estonia](#)
  - [Finland](#)
  - [France](#)
  - [Germany](#)
  - [Greece](#)
  - [Ireland](#)
  - [Italy](#)
  - [Latvia](#)
  - [Lithuania](#)
  - [Luxembourg](#)
  - [Malta](#)
  - [Monaco](#)
  - [Netherlands](#)
  - [New Zealand](#)
  - [Portugal](#)
  - [San Marino](#)
  - [Slovak Republic](#)
  - [Slovenia](#)
  - [Spain](#)
  - [Sweden](#)
  - [United States](#)
- [Character Sets](#)
- [Public Certificate Policy](#)
- [Approving our IP Addresses](#)
  - Alternative to IP approved lists
- [Tax Rates](#)
  - Who does this impact?
  - Best practices for integrating
  - Definitions of jurisdiction
  - Tax Tables
- [Security Requirements](#)
  - Accessing Bank Details



# Developer docs

[Mise en route](#) [Référence API](#) [Demandes de facturation](#)

Guides ▾

[Partner Guide](#) [Embed Guide](#) [CLI Reference](#)

[Créer un compte sandbox](#)

- [Getting Started](#)
- [API Reference](#)
- [Billing Requests](#)
- [Create a sandbox account](#)
- [Aperçu](#)
  - [Systèmes de prélèvement automatique pris en charge](#)
  - [Mise en route](#)
  - [Restrictions](#)
    - [Payment page restrictions](#)
    - [Creditor management restrictions](#)
  - [Anatomie](#)
    - [Mandate setup](#)
    - [Payment creation](#)
    - [How timings work](#)
  - [Compatibilité inverse](#)
  - [Journal des modifications](#)
  - [Bibliothèques client](#)
- [Utilisation de l'API](#)
  - [Faire des demandes](#)
    - [Base URLs](#)
    - [Authentication](#)
    - [Versions](#)
    - [MIME Types](#)
    - [PUT, PATCH et POST](#)
    - [Rate Limiting](#)
    - [Timeouts](#)
    - [Idempotency Keys](#)
    - [Optional properties](#)
  - [Dates et horaires](#)
  - [Pagination du curseur](#)

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- [Options](#)
- [Response](#)
- [Codes de réponse](#)
- [Simulateurs de scénarios](#)
- [Erreurs](#)
  - [Error types](#)
  - [GoCardless errors](#)
  - [Invalid API usage errors](#)
  - [Invalid state errors](#)
  - [Validation errors](#)
- [Demandes de facturation](#)
  - [Autorisations bancaires](#)
    - [Create a Bank Authorisation](#)
    - [Get a Bank Authorisation](#)
  - [Demandes de facturation](#)
    - [Create a Billing Request](#)
    - [Collect customer details](#)
    - [Collect bank account details](#)
    - [Confirm the payer details](#)
    - [Fulfil a Billing Request](#)
    - [Cancel a Billing Request](#)
    - [List Billing Requests](#)
    - [Get a single Billing Request](#)
    - [Notify the customer](#)
    - [Trigger fallback](#)
    - [Change currency](#)
    - [Select institution for a Billing Request](#)
  - [Flux de demandes de facturation](#)
    - [Create a Billing Request Flow](#)
    - [Initialise a Billing Request Flow](#)
  - [Modèles de demande de facturation](#)
    - [List Billing Request Templates](#)
    - [Get a single Billing Request Template](#)
    - [Create a Billing Request Template](#)
    - [Update a Billing Request Template](#)
  - [Demande de facturation avec actions](#)
    - [Create a Billing Request with Actions](#)
  - [Institutions](#)
    - [List Institutions](#)
    - [List institutions for Billing Request](#)
- [Points de terminaison principaux](#)
  - [Soldes](#)
    - [List balances](#)
  - [Détails du compte bancaire](#)
    - [Get encrypted bank details](#)
  - [Blocs](#)
    - [Create a block](#)
    - [Get a single block](#)
    - [List multiple blocks](#)
    - [Disable a block](#)
    - [Enable a block](#)
    - [Create blocks by reference](#)
  - [Créanciers](#)
    - [Create a creditor](#)
    - [List creditors](#)
    - [Get a single creditor](#)
    - [Update a creditor](#)
  - [Comptes bancaires des créanciers](#)
    - [Create a creditor bank account](#)
    - [List creditor bank accounts](#)
    - [Get a single creditor bank account](#)
    - [Disable a creditor bank account](#)
    - [Validate Creditor Bank Account](#)
  - [Taux de change](#)
    - [List exchange rates](#)
  - [Clients](#)
    - [Create a customer](#)
    - [List customers](#)
    - [Get a single customer](#)
    - [Update a customer](#)
    - [Remove a customer](#)
  - [Comptes bancaires clients](#)
    - [Create a customer](#)
    - [List customer bank accounts](#)
    - [Get a single customer bank account](#)
    - [Disable a customer bank account](#)
  - [Notifications clients](#)
    - [Handle a notification](#)
  - [Événements](#)
    - [List events](#)
    - [Get a single event](#)
    - [Reconciling Payments](#)
  - [Exportations](#)
    - [Get a single export](#)
    - [List exports](#)
  - [Horaires d'installation](#)

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- [Create \(with dates\)](#)
- [Create \(with schedule\)](#)
- [List instalment schedules](#)
- [Get a single instalment schedule](#)
- [Update an instalment schedule](#)
- [Cancel an instalment schedule](#)
- [Logos](#)
  - [Create a logo associated with a creditor](#)
- [Mandats](#)
  - [Create a mandate](#)
  - [List mandates](#)
  - [Get a single mandate](#)
  - [Update a mandate](#)
  - [Cancel a mandate](#)
  - [Reinstate a mandate](#)
- [Mandat Importations](#)
  - [Create a new mandate import](#)
  - [Get a mandate import](#)
  - [Submit a mandate import](#)
  - [Cancel a mandate import](#)
- [Obliger les entrées d'importation](#)
  - [Add a mandate import entry](#)
  - [List all mandate import entries](#)
- [Limites de solde négatif](#)
  - [List negative balance limits](#)
- [Paiements sortants](#)
  - [Create an outbound payment](#)
  - [Create a withdrawal outbound payment](#)
  - [Cancel an outbound payment](#)
  - [Approve an outbound payment](#)
  - [Get an outbound payment](#)
  - [List outbound payments](#)
  - [Update an outbound payment](#)
- [Autorisations des payeurs](#)
  - [Get a single Payer Authorisation](#)
  - [Create a Payer Authorisation](#)
  - [Update a Payer Authorisation](#)
  - [Submit a Payer Authorisation](#)
  - [Confirm a Payer Authorisation](#)
- [Thèmes du payeur](#)
  - [Create a payer theme associated with a creditor](#)
- [Paiements](#)
  - [Create a payment](#)
  - [List payments](#)
  - [Get a single payment](#)
  - [Update a payment](#)
  - [Cancel a payment](#)
  - [Retry a payment](#)
- [Paiements](#)
  - [List payouts](#)
  - [Get a single payout](#)
  - [Update a payout](#)
- [Articles de paiement](#)
  - [Get all payout items in a single payout](#)
- [Rediriger les flux](#)
  - [Create a redirect flow](#)
  - [Get a single redirect flow](#)
  - [Complete a redirect flow](#)
- [Remboursements](#)
  - [Create a refund](#)
  - [List refunds](#)
  - [Get a single refund](#)
  - [Update a refund](#)
- [Simulateurs de scénarios](#)
  - [Simulate a scenario](#)
- [Identifiants de schéma](#)
  - [Create a scheme identifier](#)
  - [List scheme identifiers](#)
  - [Get a single scheme identifier](#)
- [Abonnements](#)
  - [Recurrence Rules](#)
  - [Rolling dates](#)
  - [Create a subscription](#)
  - [List subscriptions](#)
  - [Get a single subscription](#)
  - [Update a subscription](#)
  - [Pause a subscription](#)
  - [Resume a subscription](#)
  - [Cancel a subscription](#)
- [Taux d'imposition](#)
  - [List tax rates](#)
  - [Get a single tax rate](#)
- [Mandats transférés](#)
  - [Get updated creditor](#)
- [Détails de vérification](#)
  - [Create a verification](#)
  - [List verification details](#)

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- [Webhooks](#)
  - [List webhooks](#)
  - [Get a single webhook](#)
  - [Retry a webhook](#)
- [Actions événementielles](#)
  - [Demande de facturation](#)
    - [created](#)
    - [flow\\_created](#)
    - [flow\\_visited](#)
    - [flow\\_exited](#)
    - [collect\\_customer\\_details](#)
    - [select\\_institution](#)
    - [collect\\_bank\\_account](#)
    - [payer\\_details\\_confirmed](#)
    - [bank\\_authorisation\\_visited](#)
    - [bank\\_authorisation\\_authorised](#)
    - [bank\\_authorisation\\_denied](#)
    - [bank\\_authorisation\\_expired](#)
    - [bank\\_authorisation\\_failed](#)
    - [fulfilled](#)
    - [cancelled](#)
    - [choose\\_currency](#)
    - [collect\\_amount](#)
    - [payer\\_geo\\_blocked](#)
    - [failed](#)
  - [Créancier](#)
    - [updated](#)
    - [new\\_payout\\_currency\\_added](#)
    - [account\\_auto\\_frozen](#)
    - [account\\_auto\\_frozen\\_reverted](#)
    - [bounced\\_payout](#)
  - [Export](#)
    - [started](#)
    - [completed](#)
    - [failed](#)
  - [Calendrier des versements](#)
    - [created](#)
    - [creation\\_failed](#)
    - [cancelled](#)
    - [errored](#)
    - [resumed](#)
    - [completed](#)
  - [Mandat](#)
    - [created](#)
    - [customer\\_approval\\_granted](#)
    - [customer\\_approval\\_skipped](#)
    - [active](#)
    - [cancelled](#)
    - [failed](#)
    - [transferred](#)
    - [expired](#)
    - [submitted](#)
    - [resubmission\\_requested](#)
    - [reinstated](#)
    - [replaced](#)
    - [consumed](#)
    - [blocked](#)
  - [Paiement sortant](#)
    - [created](#)
    - [cancelled](#)
    - [failed](#)
    - [set\\_to\\_pending\\_approval](#)
    - [set\\_to\\_executing](#)
    - [executed](#)
  - [Autorisation du payeur](#)
    - [completed](#)
    - [failed](#)
  - [Paiement](#)
    - [created](#)
    - [customer\\_approval\\_granted](#)
    - [customer\\_approval\\_denied](#)
    - [submitted](#)
    - [confirmed](#)
    - [chargeback\\_ca...](#)
    - [paid\\_out](#)
    - [late\\_failure\\_se...](#)
    - [chargeback\\_se...](#)
    - [surcharge\\_fee](#)
    - [failed](#)
    - [charged\\_back](#)
    - [cancelled](#)
    - [resubmission\\_r...](#)
    - [sds\\_settlement](#)
  - [Payout](#)
    - [paid](#)
    - [fx\\_rate\\_confirm...](#)
    - [tax\\_exchange\\_rates\\_confirmed](#)

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- [Remboursement](#)
  - [created](#)
  - [failed](#)
  - [paid](#)
  - [refund\\_settled](#)
  - [funds\\_returned](#)
- [Identifiant du schéma](#)
  - [activated](#)
- [Abonnement](#)
  - [created](#)
  - [customer\\_approval\\_granted](#)
  - [customer\\_approval\\_denied](#)
  - [amended](#)
  - [payment\\_created](#)
  - [cancelled](#)
  - [finished](#)
  - [paused](#)
  - [resumed](#)
  - [scheduled\\_pause](#)
  - [scheduled\\_pause\\_cancelled](#)
  - [scheduled\\_resume](#)
- [Succès+](#)
  - [Tentatives intelligentes](#)
  - [Activation des tentatives intelligentes](#)
    - [Individual Payments](#)
    - [Instalment Schedules](#)
    - [Subscriptions](#)
  - [Gestion des échecs](#)
  - [Règles de nouvelle tentative](#)
- [Points de terminaison d'aide](#)
  - [Bilan de santé](#)
  - [Recherches de détails bancaires](#)
    - [Perform a bank details lookup](#)
  - [PDF du mandat](#)
    - [Create a mandate PDF](#)
- [Appendice](#)
  - [Webhooks](#)
    - [Overview](#)
    - [Signing Webhooks](#)
    - [Examples](#)
  - [Flux JavaScript](#)
    - [Customer Bank Account Tokens](#)
    - [Create a customer bank account token](#)
    - [Intercepting form submission with the JavaScript library](#)
    - [Get a single customer bank account token](#)
  - [OAuth](#)
    - [The OAuth flow](#)
    - [Creating an OAuth app](#)
    - [Building an authorisation link](#)
    - [Handling the redirect back to you](#)
    - [Exchanging an authorisation code for an access token](#)
    - [Making requests](#)
    - [Receiving webhooks](#)
    - [App Fees](#)
    - [Looking up an access token](#)
    - [Disconnecting a user from your app](#)
  - [Exigences de conformité](#)
  - [Détails de la banque locale](#)
    - [IBAN](#)
    - [UK](#)
    - [Australia](#)
    - [Austria](#)
    - [Belgium](#)
    - [Canada](#)
    - [Cyprus](#)
    - [Denmark](#)
    - [Estonia](#)
    - [Finland](#)
    - [France](#)
    - [Germany](#)
    - [Greece](#)
    - [Ireland](#)
    - [Italy](#)
    - [Latvia](#)
    - [Lithuania](#)
    - [Luxembourg](#)
    - [Malta](#)
    - [Monaco](#)
    - [Netherlands](#)
    - [New Zealand](#)
    - [Portugal](#)
    - [San Marino](#)
    - [Slovak Republic](#)
    - [Slovenia](#)
    - [Spain](#)
    - [Sweden](#)
    - [United States](#)

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- [Ensembles de caractères](#)
- [Politique de certification publique](#)
- [Approuver nos adresses IP](#)
  - [Alternative to IP approved lists](#)
- [Taux d'imposition](#)
  - [Who does this impact?](#)
  - [Best practices for integrating](#)
  - [Definitions of jurisdiction](#)
  - [Tax Tables](#)
- [Exigences de sécurité](#)
  - [Accessing Bank Details](#)

# Référence API

## Aperçu

Ceci est la documentation de l'API GoCardless.

Si vous êtes simplement en train d'explorer ou si vous souhaitez commencer à créer une intégration, consultez notre manuel de démarrage [guide](#) pour une introduction étape par étape avec PHP, .Exemples de code NET, Java, Ruby et Python

À des fins de test, nous vous recommandons [créez un compte sandbox](#).

## Systèmes de prélèvement automatique pris en charge

L'API GoCardless prend actuellement en charge les schémas de prélèvement automatique suivants :

Schéma	Pays
Bacs	Royaume-Uni et îles Anglo-Normandes (inc. Jersey et Guernesey) + Île de Man + Gibraltar (le client doit sélectionner le Royaume-Uni comme pays)
Noyau SEPA*	Îles Åland Andorre Autriche Belgique Chypre Estonie Finlande France Guyane française Allemagne Grèce Guadeloupe Irlande Italie Lettonie Lituanie Luxembourg Malte Martinique Mayotte Monaco Pays-Bas Portugal Réunion Saint-Martin Slovaquie Slovénie Espagne Saint Barthélémy Saint-Martin Saint Pierre et Miquelon Cité du Vatican
ACH	États-Unis
Autogire	Suède
Service de paris	Danemark
BECS	Australie
BECS NZ	Nouvelle-Zélande
COUSSIN	Canada
Payer pour**	Australie
Paiements plus rapides***	UK & Channel Islands (inc. Jersey & Guernsey) + Isle of Man + Gibraltar (The customer needs to select UK as the country)

**Note:** There are practical issues with SEPA Direct Debit in some countries that mean we would not recommend using it to collect payments from end customers in those markets. Local implementation can cause problems in the Baltic states in particular - please [contact us](#) for more information.

\* Collection from all non-Eurozone SEPA countries is also supported through the API, but since only a small percentage of bank accounts in these countries are reachable they have not been listed above.

\*\* PayTo although not technically a Direct Debit scheme can be used as an alternative to BECS.

\*\*\* Faster Payments is not a Direct Debit scheme, but an alternative to BACS that can be used for instant and variable recurring payments.

## Getting Started

There are three key parts to an integration with the API - you can do one, two or all three of them:

- Setting up mandates with your customers' bank accounts
- Collecting payments against your mandates
- Staying up-to-date with webhooks

To get started with the API, check out our "[getting started](#)" guide, with copy and paste code samples in PHP, Ruby and Python guiding you through your integration from start to finish.

## Restrictions

Whilst the entire GoCardless API

### Payment page restrictions

Unless your payment pages have 1 accounts and mandates.

The following endpoints are there

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[API](#) to create customers, bank



Consultez notre politique de confidentialité

- **Mandate:** Create, Reinstate
- **Javascript flow:** All endpoints

Please [get in touch](#) to discuss having your payment pages approved.

## Creditor management restrictions

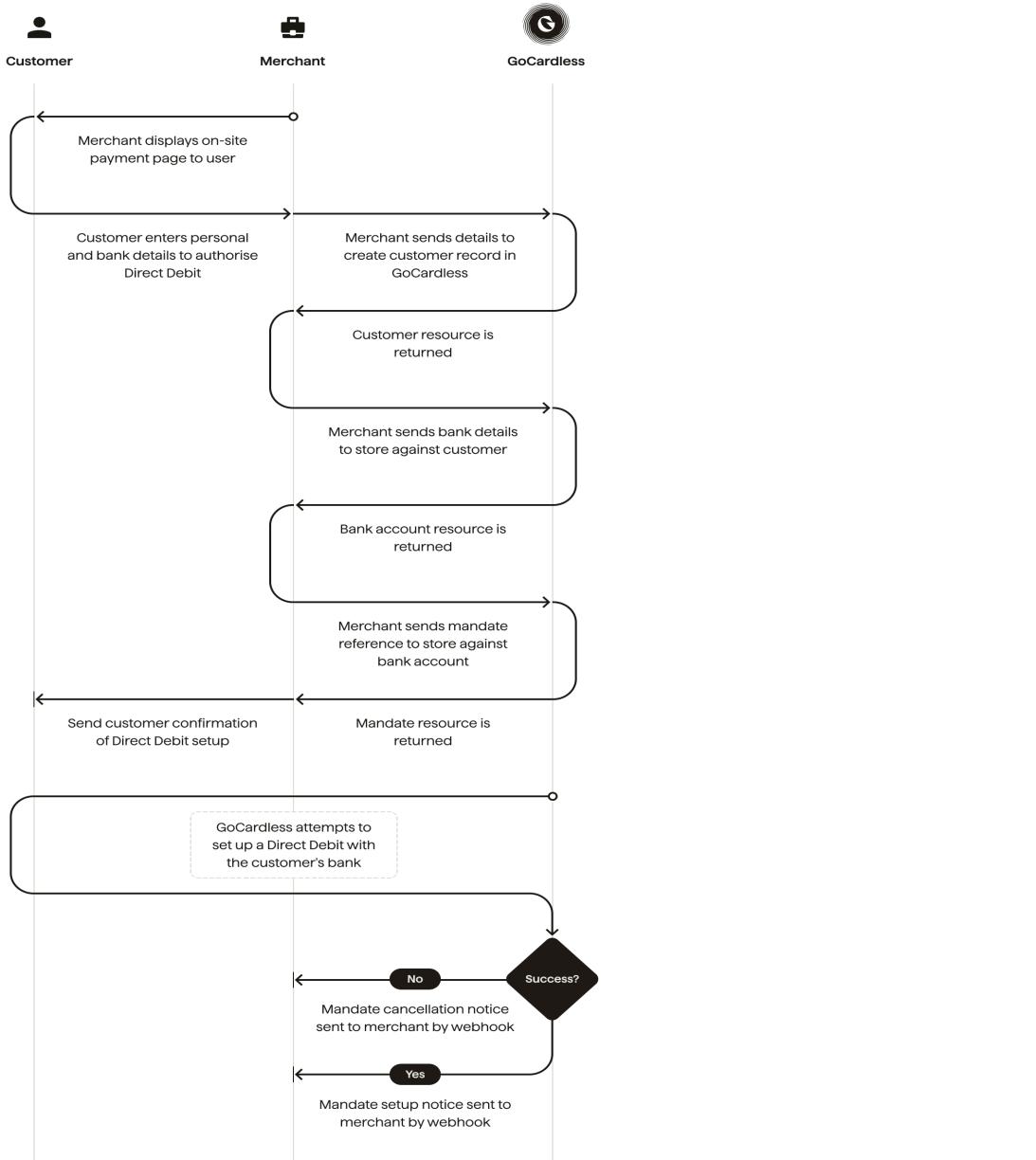
Unless your account has previously been approved as a whitelabel partner you may only collect payments on behalf of a single creditor. The following endpoints are therefore restricted:

- **Creditors:** Create

## Anatomy

The following state diagrams explain the flows of mandate setup to setup customers on a Direct Debit and payment creation, to collect payment against said mandate. There's also a short section on how timings work.

### Mandate setup



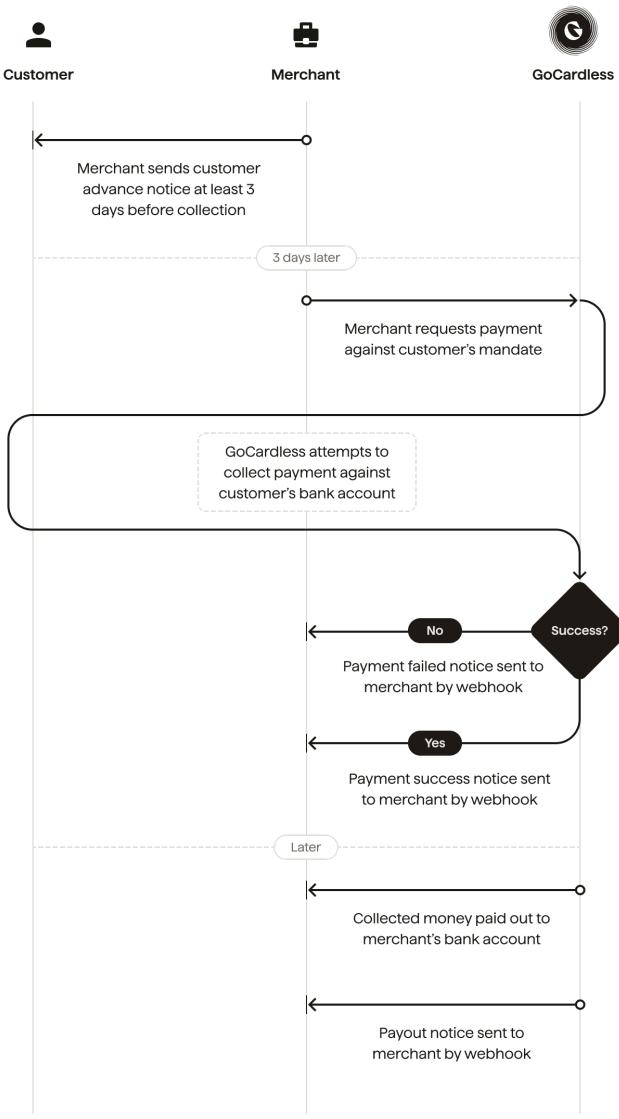
### Payment creation

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité



## How timings work

Direct debit payments don't process instantly, and work differently to card payments.

### For the UK:

- If a Direct Debit mandate is already in place, payment is collected 2 working days after submission, can be considered 95% confirmed 3 working days after submission, and 100% confirmed after 4 working days.
- If a Direct Debit mandate needs to be created, payment is collected 4 working days after submission, can be considered 95% confirmed 5 working days after submission, and 100% confirmed after 6 working days.

There's also a requirement to notify your customers 3 working days before payment leaves their account. See the [notifications section](#) for more detail.

For more information on UK timings, see our [detailed guide](#).

### For ACH:

Standard ACH can take 4-5 business days to receive payment, while Faster ACH through GoCardless will take 2-3 business days.

For more information on ACH timings, see our [detailed guide](#).

### For Autogiro:

- New mandates take up to 6 interbank business days before payments can be submitted under them.
- For all payment collections, you must submit the collection to the banks 2 interbank business days before the payment due date.

You will need to notify your custo

For more information on Autogiro

### For BECS:

- For all payment collections.

For more information on BECS ti

### For BECS NZ:

- For all payment collections.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

For more information on BECS NZ timings, see our [detailed guide](#).

#### For Betalingsservice:

- New mandates must be created at least one working day before payments can be created.
- For all payment collections, you must submit the collection 8 working days before the end of the month preceding the month in which you wish to take the payment

For more information on Betalingsservice timings, see our [detailed guide](#).

#### For PAD:

- For all payment collections, you must submit the collection to the banks 3 interbank business days before the payment due date.

For more information on PAD timings, see our [detailed guide](#).

#### For PayTo:

- There is no advance notice period and payments can be taken almost instantaneously.

#### For Faster Payments:

- There is no advance notice period and payments can be taken almost instantaneously.

#### For SEPA Core:

- For all payment collections, you must submit the collection to the banks 1 interbank business days before the payment due date.

You will need to notify your customers 2 working days in advance of the payment being collected or as soon as the payment is created if the payment is created less than 2 days before it is collected.

For more information on SEPA timings, see our [detailed guide](#).

In all cases, you can still test these as if they were instant - for more detail, see the section on [testing your integration](#).

## Backwards Compatibility

The following changes are considered backwards compatible:

- Adding new API endpoints
- Adding new properties to the responses from existing API endpoints
- Reordering properties returned from existing API endpoints
- Adding optional request parameters to existing API endpoints
- Altering the format or length of IDs
  - These strings will never exceed 255 characters, but you should be able to handle anything up to that length
- Altering the message attributes returned by validation failures / other errors
- The number of and duration between retries for [failed webhooks](#)
- The behaviour of [Scenario Simulators](#)
- Sending webhooks for new event types

## Changelog

9th May 2025

Removed [Refund](#) API limit of 25 refunds per payment.

26th March 2025

Added a new endpoint `/balances` that returns balances for a creditor

27th January 2025

Exposed processing errors in [Mandate Import Entries](#) API:

- Added status filter parameter.
- Added processing\_errors response field.

2nd December 2024

Added ACH and PAD-specific API fields:

- Added `mandate_request[consent_type]` parameter to [Billing Request](#) creation.
- Added `constraints[payment_method]` parameter to [Billing Request](#) creation.
- Added `subscription_request` parameter to [Billing Request](#) creation.
- Added `instalment_schedule_request` parameter to [Billing Request](#) creation.

8th January 2024

Added API fields for Faster ACH:

- Added `payment[faster_ach]` parameter to [Payment](#) creation.
- Added `payment[faster_ach]` response field to [Payment](#) responses.
- Added `mandate[next_possible_standard_ach_charge_date]` response field to [Mandate](#) responses.

18th July 2023

Updated our [public certificate policy](#).

21 June 2023

Added the `resource_metadata`

16 March 2023

Added `show_success_redirect`

1st March 2023

Changed the [Payout Items](#)

22nd February 2023

Made the [Billing Requests](#) to infer the language from the `language` parameter.

8th February 2023

Added `billing_request[push]`

13th January 2023

Added the `verified_at` timestamp

23rd December 2022

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Updated our [public certificate policy](#).

20th December 2022

Added `authorisation_source` parameter to the [Mandate](#) creation API and `mandate_request[authorisation_source]` parameter to the [BillingRequest](#) creation API. This field is required for offline mandates.

13th December 2022

Added `creditor_type` parameter to the [Creditors](#) API. This field represents the type of business of the creditor. It will now be required when creating a creditor and it will be returned when fetching creditors.

9th December 2022

Added `language` parameter to [Billing Request Flow](#) creation.

11th November 2022

Added `mandate_request[description]` and `mandate_request[constraints]` parameter to [Billing Request](#) creation. Constraints restricts the kind of payment that can be taken against a mandate.

1st November 2022

[Payout Items](#) are no longer ordered by type. The order of Payout Items for a given Payout is still consistent between requests.

5th August 2022

Added `payer_requested_dual_signature` parameter to [Billing Request](#) action confirm payer details.

21st July 2022

Added `prefilled_customer` and `prefilled_bank_account` parameters to [Billing Request](#) creation.

14th July 2022

Added `mandate_request[reference]` parameter to [Billing Request](#) creation.

10th May 2022

Autogiro scheme is now available in Billing Request flow. Documentaion can be found [here](#)

4th May 2022

[Event](#) creation is now an asynchronous process. This means it can take some time between an action occurring and its corresponding event being made available in API responses.

26th January 2022

Added support for filtering [events](#) by [instalment schedule](#) in the events [List API](#).

31st December 2021

Added support for Sepa Credit Transfer and Sepa Instant Credit Transfer, the Instant Bank Pay schemes inside Billing Request.

18th November 2021

Added [block](#) endpoints. These endpoints allow to create a block based on payers bank\_account, email and email\_domain, view the blocked items.

5th August 2021

Fixed reason codes sent for BECS so that they are consistent with the [documentation](#). Eg. `direct_debit_not_enabled` events will be sent with Reason code 2 instead of 'RETURNS-2', as per documentation.

30th March 2021

Added [Scenario Simulators](#) endpoint. These allow manual testing of certain flows in the Sandbox environment

26th March 2021

Added a new resource called [Billing Requests](#). This is our new API dedicated to building custom payment pages. It encapsulates the creation of customer, bank account and mandate under this single resource.

28th January 2021

Added [webhook](#) endpoints. These allow you to view and retry webhooks that we have sent to you. As part of this we began including `webhook_id` inside the meta key in the webhooks we send.

30th October 2020

Added a new resource called [payer\\_authorisations](#). This is our new API dedicated to building custom payment pages. It encapsulates the creation of customer, bank account and mandate under this single resource.

12th August 2020

Added a new event called `payer_authorisation_completed` for an upcoming resource type called `payer_authorisation`.

14th July 2020

Added support for applying tax to transaction and surcharge fees.

- Added `taxes` to [payout\\_items](#).
- Added `tax_currency` to [payout](#).
- Added [tax\\_rates endpoint](#).
- Added a payout [tax\\_exchange\\_rates\\_confirmed webhook](#) to know when the exchange rate has been finalised for all fees in the payout.

9th July 2020

Allow changing `retry_if_possible` when [updating a subscription](#).

27th May 2020

Added a `not_retried_reason` field to the event details for failed payment [event](#).

7th May 2020

Updated the type of `bank_account_type` from string to enum. `bank_account_type` is referenced in `creditor_bank_accounts`, `customer_bank_account_tokens`, `customer_bank_accounts` and `mandate_pdfs` endpoint.

20th April 2020

Added `prefilled_bank_account` attribute in the redirect flow create endpoint to allow the integrator to prefill the `account_type` field.

20th April 2020

Added metadata to payouts, and added new [update\\_payouts](#) route so that integrators can use it.

6th April 2020

We are updating the wording of the documentation for `prefilled_customer[region]` attribute in redirect flow create endpoint.

26th March 2020

We have added the ability to capture metadata attribute as part of redirect flow create endpoint.

17th March 2020

We've added the ability to [pause](#) and [resume](#) subscriptions.

19th February 2020

We've optimised our webhook batching to become slightly more effective. This means each webhook will have on-average-slightly more events per request than we have sent previously.

18th February 2020

Added new [events](#) for credi

- `updated`
- `new_payout_currency`

6th February 2020

Added support for [Success](#):

- Added a new `retry_i`
- Added a `will_attemp`

6th February 2020

Added new [disconnected w](#)

29th January 2020

Added 3 fields to the [Credi](#)

28th January 2020

Added support for instalment schedules.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

ment\_pages\_enabled.

- Added documentation for new [Instalment Schedules](#) endpoint.

27th January 2020

Marked the `phone_number` field in [Customers](#) as optional.

19th November 2019

Added support for international payments.

- Added fx to [Payments](#), [Payouts](#) & [Refunds](#)
- Added support for [surcharge fees](#).
- Added new [surcharge\\_fee\\_type](#) for [Payout Items](#).
- Added payment webhooks for surcharge fee events (`surcharge_fee_credited` and `surcharge_fee_debited`)

23rd July 2019

Added new [customer removal API](#).

21st February 2019

Added new `payment_autoretried` event cause for the `payment_resubmission_requested` [action](#).

19th December 2018

Added the ability to [handle subscription created emails](#).

5th December 2018

Added support for PAD, the Direct Debit scheme in Canada, in beta.

16th October 2018

Added new [event](#) for refund `failed` (`refund_failed`).

11th September 2018

Added support for BECS NZ, the Direct Debit scheme in New Zealand, in beta.

30th August 2018

Add `links[default_aud_payout_account]` as an updatable field to the [creditors](#) API

22nd August 2018

Removed `can_create_refunds` as an updatable field from the [creditors](#) API

17th August 2018

Added the ability to [handle payment created and mandate created emails](#).

12th July 2018

Added customer address fields to the [mandate PDFs](#) API.

27th June 2018

Added support for Betalingsservice, the Direct Debit scheme in Denmark, in beta.

22nd May 2018

Added `app_fee` field to responses from the [subscriptions](#) API.

17th May 2018

Took the [payout items](#) API out of beta.

Improved the documentation of the [mandate PDFs](#) API, clarifying what languages are supported for each scheme and documenting that the default language is English.

15th May 2018

Added the maximum [payment](#) and [refund](#) reference lengths for Becs.

26th April 2018

Added the [mandate imports](#) API, allowing partners to more easily migrate mandates to GoCardless.

20th April 2018

Added information on [our public certificate policy](#).

11th April 2018

Added support for BECS, the Direct Debit scheme in Australia, in beta.

6th March 2018

Added contact email address to responses from the [OAuth introspect API](#) and [access token API](#).

13th February 2018

Added support for [multiple redirect URIs per partner app](#).

26th January 2018

Added the new [OAuth introspect API](#), allowing partners to check if access tokens are valid and find out more about them

Added the new [OAuth revoke API](#), allowing partners to disconnect users from their app, invalidating their access token

15th January 2018

Added additional characters to the list of allowed characters for SEPA mandate and payment references

1st December 2017

Added read-only `can_create_refunds` field to responses from the [creditors](#) API.

30th November 2017

Renamed `exceeded_max_amendments` error on subscriptions to `number_of_subscription_amendments_exceeded` when [updating a subscription](#).

16th November 2017

Added filtering by `reference` to the [payouts](#) API.

14th November 2017

Added filtering by `status` to the [subscriptions](#) API.

13th November 2017

Added the new [payout items](#) API, allowing you to view, on a per-payout basis, the credit and debit items that make up that payout's amount.

25th October 2017

Allow changing `amount` when [updating a subscription](#).

23rd October 2017

Added new [event](#) for subscription amended action.

18th September 2017

Added `confirmation_url` to the [redirect flows](#) API.

19th June 2017

Added the `Origin` header to webhooks, specifying the GoCardless environment that a webhook was sent from (<https://api.gocardless.com> for live, <https://api-sandbox.gocardless.com> for sandbox).

19th May 2017

Added `verification_status`

11th May 2017

Deprecated `end_date` in the

17th March 2017

Added new `mandate_expire`

13th March 2017

Added `app_fee` to the [subsc](#)

2nd March 2017

Added `prefilled_customer`

20th February 2017

Added `scheme_identifiers`

2nd February 2017

Added new [event](#) for mandate replaced action.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

2nd August 2016

Added deducted\_fees to the [payments](#) API.

7th July 2016

Added payments\_require\_approval to the [mandates](#) API.

11th February 2016

Added arrival\_date to the [payments](#) API.

21st January 2016

Restricted specifying payment references on Bacs payments to only those organisations using their own Service User Number.

6th January 2016

Added filtering by created\_at to the [customer bank accounts](#) API.

20th October 2015

Added filtering by created\_at to the [mandates](#) API.

16th October 2015

Added filtering by created\_at to the [payments](#) API.

30th September 2015

Added support for [app fees](#) to the OAuth API, in beta.

29th September 2015

Added support for Autogiro, the Direct Debit scheme in Sweden, in beta.

16th September 2015

Added request\_pointer field to validation errors, in the form of [JSON pointer](#). See [validation errors](#) for more details.

10th September 2015

Added reference to the [refund](#) API. This sets a reference which will appear with the refund on the customer's bank statement.

17th August 2015

Added language to the [customer](#) API. This sets the language of notification emails sent by GoCardless to the customer. Defaults to the native language of the customer's country.

2nd August 2015

Added support for SEPA COR1, a faster version of SEPA Direct Debit available in Spain, Germany and Austria. COR1 will be used automatically if possible when creating new SEPA mandates, unless you specify the scheme attribute explicitly. The returned scheme attribute will be sepa\_cor1.

29th July 2015

Added customer\_bank\_account and customer to the [complete redirect flow](#) endpoint returned properties.

23rd July 2015

Removed support for API version 2014-11-03.

22nd July 2015

Added bic to the [bank details lookups](#) endpoint returned properties.

6th July 2015

Released API version 2015-07-06. See [blog post](#) for more details and upgrade guide.

- Removes Helpers endpoint from the API
- Renames start\_at and end\_at on subscriptions to start\_date and end\_date
- Enforces date format (not datetime) for payment charge\_date

2nd July 2015

Renamed the beta modulus checks endpoint to [bank details lookups](#) and took it out of beta.

30th June 2015

Added [mandate PDFs](#) endpoint, which will replace the previous mandate helper endpoint and the ability to request a PDF from the "get a single mandate" endpoint in a future version.

26th June 2015

Added top level modulus checks endpoint, which will replace the previous modulus check helper endpoint in a future version.

15th June 2015

Updated MIME type negotiation to 406 if a bad Accept header is provided.

9th June 2015

Added new [events](#) for payment created, subscription created, and refund paid.

29th April 2015

Released API version 2015-04-29. See [blog post](#) for more details and upgrade guide.

- Removes Roles and Users from the API
- Replaces Api Keys with Access Tokens
- Replaces Publishable Api Keys with Publishable Access Tokens
- Removes explicit sort\_code field from bank account creation APIs in favour of [local details](#)
- Removes Webhook-Key-Id header from webhooks we send

## Client Libraries

We also provide client libraries to make your integration easier.

Currently we have the following libraries available:

- [GoCardless Pro Ruby](#)
- [GoCardless Pro Python](#)
- [GoCardless Pro Java](#)
- [GoCardless Pro PHP](#)
- [GoCardless .NET](#)
- [GoCardless Node.js](#)
- [GoCardless Pro Go](#)
- [GoCardless Pro Android](#)
- [GoCardless Pro iOS](#)

If there's another language you'

## API Usage

### Making Requests

#### Base URLs

The base URLs for the GoCardles

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

dbox

The API is only available over HTTPS. Attempting to access the API over an unsecured HTTP connection will return a `tls_required` error.

## Authentication

After creating an access token in the dashboard, you must provide it in an Authorization request header (using the [bearer](#) authentication scheme) when making API requests.

```
Authorization: Bearer TOP_SECRET_ACCESS_TOKEN
```

If you're using our [JavaScript Flow](#), you must use a publishable access token instead.

## Versions

Every request must specify a GoCardless-Version header, with a released [API version](#).

GoCardless-Version: 2015-07-06

Currently available versions:

2015-07-06

Removed helper endpoint (replaced with new individual endpoints).

Historic versions which are no longer available:

2015-04-29

Changed auth and removed sort\_code.

2014-11-03

Changed validations on charge\_customer\_at.

2014-10-03

Enforced validation of keys passed to all endpoints.

2014-09-01

First Pro API release.

## MIME Types

All requests and responses are JSON-formatted and UTF-8 encoded.

An Accept header is required for all requests, for example:

```
Accept: application/json
```

A Content-Type and Content-Length header must be given when sending data to the API (using POST and PUT endpoints), for example:

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

Content-Length: 3495

For the Accept and Content-Type headers, you may give either the standard JSON MIME type (`application/json`), or the JSON-API variant (`application/vnd.api+json`).

You will receive an `invalid_content_type` error if you attempt to make a POST/PUT request without one of these MIME types.

In the case of a missing Content-Length header, you will receive an Error 411 (Length Required) error message. Although, the majority of HTTP clients should send the header by default.

## PUT, PATCH and DELETE

If your HTTP client or proxy doesn't support PUT or DELETE, you can instead make a POST request with the header X-HTTP-Method-Override: PUT or X-HTTP-Method-Override: DELETE.

The HTTP PATCH verb is not allowed by the API, and will result in a `method_not_allowed` error. Please use PUT to alter an existing resource.

## Rate Limiting

We apply a rate limit to all API requests, to prevent excessive numbers of simultaneous requests from an individual integrator degrading the API experience for others.

Currently, this limit stands at 1000 requests per minute. If you are making requests from a partner integration (on behalf of a merchant), the rate limit is 1000 requests per minute per merchant. The limit can always be found in the response header `ratelimit-limit`.

If your API call returns a 429 status code with a `rate_limit_exceeded` error, you have exceeded this limit. These requests can be safely retried. We supply information in the `ratelimit-remaining` and `ratelimit-reset` headers, which indicate how many requests are allowed in the current time window, and the time after which the rate limit will reset, respectively.

For example, reading response headers like these, your integration is permitted to make a further 163 requests until Thu, 03 May 2018 16:00:00 GMT, after which the limit will reset back to 1000 for the next time window:

```
ratelimit-limit: 1000
ratelimit-remaining: 163
ratelimit-reset: Thu, 03 May 2018 16:00:00 GMT
```

If you are planning to make a large number of requests per minute, your task will be

## Timeouts

Some timeouts affect HTTP requests:

- **Request Timeout** - we have a timeout of 10 seconds for each request.
- **Keepalive Timeout** - we have a timeout of 10 seconds for each connection, with no request for that connection.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

can successfully hit 1000

ions. If a connection has been

## Idempotency Keys

Some requests are dangerous to make twice. In a situation like submitting payments, retrying a request that appeared to fail (for example due to a network timeout) could lead to a payment being taken twice. This is clearly undesirable.

When creating resources, our API supports idempotent creation. If you make a request with an `Idempotency-key` header, we only allow that key to be used for a single successful request. Any requests to create a resource with a key that has previously been used will not succeed, and will respond with a `409 idempotent_creation_conflict` error, with the error including a `links.conflicting_resource_id` attribute pointing to the already-existing resource.

Our [API libraries](#) will automatically generate idempotency keys when you try to create a resource. However, you might want to generate and supply your own - for example, using an ID from your database to identify a record (for example a payment you're going to collect) will protect against not only network errors, but also accidentally doing the same thing twice from your side.

Idempotency keys may be no longer than 128 characters.

Idempotency keys are intended to prevent conflicts over short periods of time, and will not be persisted indefinitely. We guarantee that we will honour idempotency keys for at least 30 days. After this point we reserve the right to respect the keys for any further length of time and this behaviour must not be relied upon. We strongly encourage generating idempotency keys using UUIDv4 but any non-repeating unique identifier is sufficient.

## Optional properties

When making requests, optional properties should be completely omitted from the JSON when not being used. This is automatically handled in our [API libraries](#) and is only relevant if you create your own.

## Dates and Times

All timestamps are formatted as ISO8601 with timezone information. For API calls that allow for a timestamp to be specified, we use that exact timestamp. These timestamps look something like `2014-02-27T15:05:06.123Z`.

For endpoints that require dates, we expect a date string of the format `YYYY-MM-DD`, where an example would look like `2014-02-27`.

## Cursor Pagination

All list/index endpoints are ordered and paginated reverse chronologically by default.

### Options

The following options are available on all cursor-paginated endpoints:

`before`  
ID of the object immediately following the array of objects to be returned.  
`after`  
ID of the object immediately preceding the array of objects to be returned.  
`limit`  
Upper bound for the number of objects to be returned. Defaults to 50. Maximum of 500. Minimum of 1.

### Response

Paginated results are always returned in an array, and include the following meta data:

`before`  
The ID of the first resource that has been returned.  
`after`  
The ID of the last resource that has been returned.  
`limit`  
The upper bound placed on the number of objects returned. If there were not enough remaining objects in the list of data then fewer than this number will have been returned.



HTTP  
HTTP

```
GET https://api.gocardless.com/resources?after=ID789 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "meta": {
    "cursors": {
      "after": "ID456",
      "before": "ID123"
    },
    "limit": 50
  },
  "resources": [
    ...
  ]
}
```

## Response Codes

You may encounter the following

`200` OK. The request has succeeded.  
`201`

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

If you experience any issues, please let us know so we can investigate and fix the cause of the problem.



Consultez notre politique de confidentialité

Created.	The request has been fulfilled and resulted in a new resource being created. The newly created resource can be referenced by the URI(s) returned in the entity of the response, with the most specific URI for the resource given by a <code>Location</code> header field.
204	No Content. The request was successful but no body was returned.
400	Bad Request. The request could not be understood by the server, usually due to malformed syntax. The response body will contain more details in the form of an array.
401	Unauthorized. The client has not provided a valid <code>Authentication</code> HTTP header or the user making the request has been disabled.
403	Forbidden. The client has provided a valid <code>Authentication</code> header, but does not have permission to access this resource.
404	Not Found. The requested resource was not found or the authenticated user cannot access the resource. The response body will explain which resource was not found.
405	Method Not Allowed. The HTTP verb used, or resource requested is not allowed. Note that we do not allow the <code>PATCH</code> verb to be used, and <code>PUT</code> should be used instead.
406	Not Acceptable. The content type specified in the <code>Accept</code> header was not acceptable to this endpoint.
409	Conflict. The resource to be created by your request already exists.
410	Gone. The authenticated user can access the requested resource, but it has been removed. The response body will explain which resource has been removed.
415	Unsupported Media Type. The response error will explain what types are accepted.
422	Unprocessable Entity. Could not process a <code>POST</code> or <code>PUT</code> request because the request is invalid. The response body will contain more details.
426	Upgrade Required. An unsecured connection was refused. Upgrade to TLS/SSL.
429	Too Many Requests. A rate limit has been reached. The headers will explain the details of the rate limit.
500	Internal Server Error. The server encountered an error while processing your request and failed. Please retry the request. If the problem persists, please report this to the <a href="#">GoCardless support team</a> and quote the <code>request_id</code> .
504	Gateway Timeout. The request timed out and should be retried. If the problem persists, please report this to the <a href="#">GoCardless support team</a> and quote the <code>request_id</code> .

## Scenario Simulators

When you're building an integration with the API, there are some common paths you should make sure your integration handles successfully.

In the sandbox environment, we provide scenario simulators which allow you to manually trigger certain cases (like a customer cancelling their mandate or a payment failing due to lack of funds) from the Dashboard so you can test how your integration responds.

To find out more, see our [scenario simulators guide](#).

## Errors

Every error includes a `type`, the HTTP status code, a short `message`, a `documentation_url`, linking to the relevant section of this document, and a unique `request_id` which can be used to help the GoCardless support team find your error quickly.

There is also an `errors` key which is an array listing any errors that have been raised in one of two formats.

When the `type` is `validation_failed`, each error will be composed of a `field` and a `message` describing what is wrong with that field.

For all other errors, the structure will contain a `reason` and a `message` describing what the problem encountered.

There are 4 types of errors, depending on the root cause. These are:

### Error types

#### gocardless

This is an error that has occurred within GoCardless while processing your requests. In some cases, depending on the specific issue highlighted in `errors`, the request should be retried. If the problem persists, it should be reported to our [support team](#) with the `request_id`, so we can resolve the issue.

#### invalid\_api\_usage

This is an error with the request you made. It could be an invalid URL, the authentication header could be missing, invalid, or grant insufficient permissions, you may have reached your rate limit, or the syntax of your request could be incorrect. The `errors` will give more detail of the specific issue.

#### invalid\_state

The action you are trying to perform is invalid due to the state of the resource you are requesting it on. For example, a payment you are trying to cancel might already have been submitted.

#### validation\_failed

The parameters submitted via the `body` parameter indicates the exact error.

These four types should be handled by your application. You should send an alert to your development team to correct their input.

### GoCardless errors

#### internal\_server\_error

An internal error occurred while processing your request.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

ponse. The `request_pointer`

ess support; `invalid_api_usage` alert the end user and prompt

we can resolve the issue.

**request\_timed\_out**

The request did not complete within a reasonable time. In this case, the request should be retried (possibly with different parameters). If the problem persists, it should be reported to our [support team](#) with the `request_id`, so we can resolve the issue.

**query\_timed\_out**

The query for this request did not complete within a reasonable time, potentially due to being too complex. In this case, the request should be retried (possibly with different parameters). If the problem persists, it should be reported to our [support team](#) with the `request_id`, so we can resolve the issue.

**Invalid API usage errors****invalid\_type**

The `errors` key may also hold an array of type errors if the JSON you sent was incorrectly typed. These are in the same format as validation errors (with a message and field per error). A type error will also be returned if you include any additional, unknown parameters.

**path\_not\_found**

The path was not recognised. Check that you spelled the resource name correctly, and that the URL is formatted correctly.

**resource\_not\_found**

The ID in the request was not found in our database.

**link\_not\_found**

One of the `link[resource]` IDs in the request was not found. Your integration should ensure that end users can only use existing resources.

**unauthorized**

Your username/password was not recognised.

**forbidden**

You were authenticated, but you do not have permission to access that resource.

**feature\_disabled**

You are trying to use a feature which hasn't been enabled on your account. Please contact support if you would like to enable it.

**not\_acceptable**

The content type specified in your `Accept` header was not acceptable to this endpoint.

**request\_entity\_too\_large**

The body of your request is too large.

**unsupported\_media\_type**

The API communicates using JSON only. Make sure that your `Accept` header permits JSON, and your `Content-Type` header is [supported](#), if you are sending JSON data (e.g. with a `POST` or `PUT` request).

**rate\_limit\_exceeded**

You have exceeded the [rate limit](#). See the included headers for when your rate limit will be reset.

**access\_token\_not\_found**

No access token with the ID specified was found.

**access\_token\_not\_active**

The access token you are using has been disabled.

**access\_token\_revoked**

The access token you are using has been revoked by the user.

**missing\_authorization\_header**

No Authorization header was included in your request. See [making requests](#) for details on how to structure your authorisation header.

**invalid\_authorization\_header**

The Authorization header sent was not valid. Make sure it was constructed as described in [making requests](#).

**insufficient\_permissions**

The access token you are using does not have the right scope to perform the requested action.

**method\_not\_allowed**

The HTTP verb used is not permitted. Note that we do not allow PATCH requests, and PUT must be used to update resources.

**bad\_request**

The request syntax was incorrect.

**idempotency\_key\_too\_long**

An idempotency key was supplied for this request but exceeded the max length of this key. See [idempotency keys](#) for details on how to work with idempotency.

**invalid\_document\_structure**

The JSON sent to the server was not valid. See the examples below for more information.

**invalid\_content\_type**

When including a JSON body, make sure the Content-Type header is set to application/json.

**tls\_required**

The GoCardless API can only be accessed via HTTPS.

**missing\_version\_header**

No GoCardless-Version header was provided.

**version\_not\_found**

The GoCardless-Version specified is not supported.

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

must be sent under the name of

application/vnd.api+json.

not http://.

er.

**invalid\_filters**

The combination of filters specified in the query string of your request are not allowed. Only certain combinations of filters may be applied to each list endpoint, as documented on each endpoint.

**request\_body\_not\_allowed**

Sending a request body is not supported for the HTTP method you have used. Use query string parameters in the URL instead.

**customer\_data\_removed**

The customer has been removed and they can no longer be returned by our API. You should remove any GoCardless references to these objects in your systems.

**payout\_items\_data\_archived**

Payout items for payouts created more than 6 months ago have been archived. Please contact support if you require access to this data.

**Invalid state errors****cancellation\_failed**

The [mandate](#), [payment](#) or [subscription](#) was not in a cancellable state. It might have already been cancelled, failed, or it might be too late in the submission process to cancel. For example, payments cannot be cancelled once they are submitted to the banks.

**retry\_failed**

The [payment](#) could not be retried.

**disable\_failed**

The [customer](#) or [creditor](#) bank account could not be disabled, as it is already disabled.

**mandate\_is\_inactive**

The [payment](#) could not be created, because the [mandate](#) linked is cancelled, failed, or expired.

**mandate\_replaced**

The resource could not be created, because the [mandate](#) it links to has been replaced (for example, because the creditor has moved to a new Service User Number). The new mandate can be found through the reference to `links[new_mandate]` in the error response, or by retrieving the original [mandate](#) and checking `links[new_mandate]`.

**bank\_account\_disabled**

The [mandate](#) could not be created because the [customer bank account](#) linked is disabled.

**mandate\_not\_inactive**

The [mandate](#) could not be reinstated, because it is already being submitted, or is active.

**refund\_is\_unreachable**

The [refund](#) could not be created, because it would not reach the target bank account.

**refund\_payment\_invalid\_state**

The [refund](#) could not be created, because the [payment](#) specified is not `confirmed` or `paid_out`.

**total\_amount\_confirmation\_invalid**

The [refund](#) could not be created because the total amount refunded does not match.

**number\_of\_refunds\_exceeded**

The [refund](#) could not be created because five refunds have already been created for the given [payment](#).

**idempotent\_creation\_conflict**

The resource has not been created as a resource has already been created with the supplied idempotency key. See [idempotency keys](#) for details.

**customer\_bank\_account\_token\_used**

The [customer bank account](#) could not be created because the token given has already been used.

**billing\_request\_must\_be\_ready\_to\_fulfil**

The [billing request](#) must have no outstanding required actions. To check the actions please view the actions array in the [billing request get response](#).

**Validation errors**

The errors key may also hold an array of individual validation failures in this case, or one of the following errors.

**bank\_account\_exists**

The [customer](#) or [creditor](#) bank account you are trying to create already exists. These resources must be unique.

You should use the corresponding update endpoints to update the details on the existing bank account instead, which will be referenced as `links[customer_bank_account]` or `links[creditor_bank_account]` (as appropriate) in the error response.

**available\_refund\_amount\_insufficient**

The [refund](#) requested by the [creditor](#) could not be created for the given currency, because the creditor does not have a sufficient balance available to cover the cost of the refund.

The amount available for repayments, pending refunds, refunds will be returned in 1 month in GBP, cents in EUR).



HTTP

HTTP

POST [https://api.gocardless.com/customer\\_bank\\_accounts](https://api.gocardless.com/customer_bank_accounts)  
Content-Type: application/json

```
{
  "customer_bank_accounts": {
    "account_number": "5577991",
    "branch_code": "I'm not a sort code",
    "account_holder_name": "Frank Osborne",
  }
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

such indicators are in-flight  
cy. The amount available for  
ition for the currency (e.g. pence

```

"country_code": "GB",
"links": {
  "customer": "CU123"
}
}

HTTP/1.1 422 (Unprocessable Entity)
Content-Type: application/json
{
  "error": {
    "documentation_url": "https://developer.gocardless.com/#validation_failed",
    "message": "Validation failed",
    "type": "validation_failed",
    "code": 422,
    "request_id": "dd50eaaf-8213-48fe-90d6-5466872efbc4",
    "errors": [
      {
        "message": "must be a number",
        "field": "branch_code",
        "request_pointer": "/customer_bank_accounts/branch_code"
      },
      {
        "message": "is the wrong length (should be 8 characters)",
        "field": "branch_code",
        "request_pointer": "/customer_bank_accounts/branch_code"
      }
    ]
  }
}

```

```

POST https://api.gocardless.com/customer_bank_accounts HTTP/1.1
Content-Type: application/json
{
  "customer_bank_accounts": {
    "account_number": "55779911",
    "branch_code": "200000",
    "account_holder_name": "Frank Osborne",
    "country_code": "GB",
    "links": {
      "customer": "CU123"
    }
  }
}

```

```

HTTP/1.1 400 (Bad Request)
Content-Type: application/json
{
  "error": {
    "message": "Invalid document structure",
    "documentation_url": "https://developer.gocardless.com/#invalid_document_structure",
    "type": "invalid_api_usage",
    "request_id": "bd271b37-a2f5-47c8-b461-040dfe0e9cb1",
    "code": 400,
    "errors": [
      {
        "reason": "invalid_document_structure",
        "message": "Invalid document structure"
      }
    ]
  }
}

```

```

POST https://api.gocardless.com/creditor_bank_accounts HTTP/1.1
Content-Type: application/json
{
  "creditor_bank_accounts": {
    "account_number": "55779911",
    "branch_code": "200000",
    "country_code": "GB",
    "set_as_default_payout_account": true,
    "account_holder_name": "Nude Wines",
    "links": {
      "creditor": "CR123"
    }
  }
}

```

```

HTTP/1.1 409 (Conflict)
Content-Type: application/json
{
  "error": {
    "message": "Bank account already exists",
    "documentation_url": "https://developer.gocardless.com/#bank_account_exists",
    "type": "validation_failed",
    "request_id": "bd271b37-a2f5-47c8-b461-040dfe0e9cb1",
    "code": 409,
    "errors": [
      {
        "reason": "bank_account_exists",
        "message": "Bank account already exists",
        "links": {
          "creditor_bank_account": "CR123"
        }
      }
    ]
  }
}

```

## Billing Requests

This section contains API endpoints.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Billing Requests help create resources that require input or action from a customer. An example of required input might be additional customer billing details, while an action would be asking a customer to authorise a payment using their mobile banking app.

Billing Requests go through a simple lifecycle, with the state exposed as the `status` field:

- `pending`, actions need completing before we can fulfil
- `ready_to_fulfil`, all required actions are complete and the integrator may fulfil this request
- `fulfilled`, the request is fulfilled and a resource has been created
- `cancelled`, request was cancelled before it can be completed (no resource will ever be created)

Integrators have a choice between building their own user experience flows to complete the required actions, or creating a Billing Request Flow against the Billing Request. Billing Request Flows are a GoCardless managed flow that knows how to process outstanding actions on the Request, helping the customer provide their required details and fulfil the request.

We advise most integrators make use of Billing Request Flows to benefit from the efforts GoCardless put into optimising conversion, adapting to a changing regulatory environment, and to reduce integration effort.

Even for integrators who want a whitelabel experience, we might advise completing some actions through a customer integration and leveraging the Billing Request Flow to complete the more complex actions.

## Bank Authorisations

Bank Authorisations can be used to authorise Billing Requests. Authorisations are created against a specific bank, usually the bank that provides the payer's account.

Creation of Bank Authorisations is only permitted from GoCardless hosted UIs (see Billing Request Flows) to ensure we meet regulatory requirements for checkout flows.

### Properties

<code>id</code>	Unique identifier, beginning with "BAU".
<code>authorisation_type</code>	Type of authorisation, can be either 'mandate' or 'payment'.
<code>authorised_at</code>	Fixed <a href="#">timestamp</a> , recording when the user has been authorised.
<code>created_at</code>	Timestamp when the flow was created
<code>expires_at</code>	Timestamp when the url will expire. Each authorisation url currently lasts for 15 minutes, but this can vary by bank.
<code>last_visited_at</code>	Fixed <a href="#">timestamp</a> , recording when the authorisation URL has been visited.
<code>qr_code_url</code>	URL to a QR code PNG image of the bank authorisation url. This QR code can be used as an alternative to providing the <code>url</code> to the payer to allow them to authorise with their mobile devices.
<code>redirect_uri</code>	URL that the payer can be redirected to after authorising the payment.

On completion of bank authorisation, the query parameter of either `outcome=success` or `outcome=failure` will be appended to the `redirect_uri` to indicate the result of the bank authorisation. If the bank authorisation is expired, the query parameter `outcome=timeout` will be appended to the `redirect_uri`, in which case you should prompt the user to try the bank authorisation step again.

Please note: bank authorisations can still fail despite an `outcome=success` on the `redirect_uri`. It is therefore recommended to wait for the relevant bank authorisation event, such as [BANK\\_AUTHORISATION\\_AUTHORISED](#), [BANK\\_AUTHORISATION\\_DENIED](#), or [BANK\\_AUTHORISATION\\_FAILED](#) in order to show the correct outcome to the user.

The BillingRequestFlow ID will also be appended to the `redirect_uri` as query parameter `id=BRF123`.

Defaults to <https://pay.gocardless.com/billing/static/thankyou>.

<code>url</code>	URL for an oauth flow that will allow the user to authorise the payment
<code>links[billing_request]</code>	ID of the <a href="#">billing request</a> against which this authorisation was created.
<code>links[institution]</code>	ID of the <a href="#">institution</a> against which this authorisation was created.

## Create a Bank Authorisation

Create a Bank Authorisation.

Relative endpoint: POST /bank\_authorisations

**Restricted:** this endpoint is restricted as the authorisation must be used within a checkout flow that meets regulatory requirements for each payment scheme. Integrators should use a Billing Request Flow instead.

If you would like to enable bank authorisations:

Parameters

`redirect_uri`

URL that the payer can be redirected to after authorising the payment.

On completion of bank authorisation, the query parameter of either `outcome=success` or `outcome=failure` will be appended to the `redirect_uri` to indicate the result of the bank authorisation. If the bank authorisation is expired, the query parameter `outcome=timeout` will be appended to the `redirect_uri`, in which case you should prompt the user to try the bank authorisation step again.

Please note: bank authorisations can still fail despite an `outcome=success` on the `redirect_uri`. It is therefore recommended to wait for the relevant bank authorisation event, such as [BANK\\_AUTHORISATION\\_AUTHORISED](#), [BANK\\_AUTHORISATION\\_DENIED](#), or [BANK\\_AUTHORISATION\\_FAILED](#) in order to show the correct outcome to the user.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

[get in touch](#).

he `redirect_uri` to indicate the result of the bank authorisation. If the bank authorisation is expired, the query parameter `outcome=timeout` will be appended to the `redirect_uri`, in which case you should prompt the user to try the bank authorisation step again.

wait for the relevant bank authorisation event, such as [BANK\\_AUTHORISATION\\_AUTHORISED](#), [BANK\\_AUTHORISATION\\_DENIED](#), or [BANK\\_AUTHORISATION\\_FAILED](#) in order to show the correct outcome to the user.

to the user.

The BillingRequestFlow ID will also be appended to the `redirect_uri` as query parameter `id=BRQ123`.

Defaults to `https://pay.gocardless.com/billing/static/thankyou`.

`links[billing_request]`  
ID of the [billing request](#) against which this authorisation was created.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/bank_authorisations HTTP/1.1
```

Content-Type: application/json

```
{
  "bank_authorisations": {
    "redirect_uri": "https://my-company.com/landing",
    "links": {
      "billing_request": "BRQ123"
    }
  }
}
```

HTTP/1.1 201 Created

Location: /bank\_authorisations/BAU123

Content-Type: application/json

```
{
  "bank_authorisations": {
    "id": "BAU123",
    "url": "https://pay-staging.gocardless.com/oauth/BAU123",
    "qr_code_url": "https://pay-staging.gocardless.com/oauth/BAU123/qr_code",
    "created_at": "2021-03-25T17:26:28.305Z",
    "authorisation_type": "payment",
    "last_visited_at": null,
    "authorised_at": null,
    "expires_at": "2021-03-25T17:41:28.000Z",
    "redirect_uri": "https://my-company.com/landing",
    "links": {
      "billing_request": "BRQ123",
      "institution": "monzo"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->bankAuthorisations()->create([
  'params' => [
    'redirect_uri' => "https://my-company.com/landing",
    'links' => [
      'billing_request' => "BRQ123"
    ]
  ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.bank_authorisations.create(params={
  "redirect_uri": "https://my-company.com/landing",
  "links": {
    "billing_request": "BRQ123"
  }
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.bank_authorisations.create(
  params: {
    redirect_uri: "https://my-company.com/landing",
    links: {
      billing_request: "BRQ123"
    }
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

BankAuthorisation bankAuthorisation =
  .withRedirectUri("https://my-company.com/landing")
  .withLinksBillingRequest("BRQ123")
  .execute();
```

JavaScript

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const bankAuthorisation = await client.bankAuthorisations.create({
  redirect_uri: "https://my-company.com/landing",
  links: {
    billing_request: "BRQ123"
  }
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var authorisationLinks = new gocardless.BankAuthorisationCreateRequest.BankAuthorisationLinks
{
  BillingRequest = "BRQ123"
};

var resp = await gocardless.BankAuthorisations.CreateAsync(
  new GoCardless.Services.BankAuthorisationCreateRequest()
  {
    RedirectUri = "https://my-company.com/landing",
    links = authorisationLinks,
  }
);

GoCardless.Resources.BankAuthorisation bankAuthorisation = resp.BankAuthorisation;

```

Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  bankAuthorisationCreateParams := gocardless.BankAuthorisationCreateParams{
    RedirectUri: "https://my-company.com/landing",
    Links: gocardless.BankAuthorisationCreateParamsLinks{
      BillingRequest: "BR123",
    },
  }

  bankAuthorisation, err := client.BankAuthorisations.Create(context, bankAuthorisationCreateParams)
}

```

## Get a Bank Authorisation

Get a single bank authorisation.

Relative endpoint: GET /bank\_authorisations/BAU123

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/bank\_authorisations/BAU123 HTTP/1.1

HTTP/1.1 200  
Content-Type: application/json

```
{
  "bank_authorisations": [
    {
      "id": "BAU123",
      "url": "https://pay-staging.gocardless.com/obauth/BAU123",
      "qr_code_url": "https://pay-staging.gocardless.com/obauth/BAU123/qr_code",
      "created_at": "2021-03-25T17:26:28.305Z",
      "authorisation_type": "payment",
      "last_visited_at": null,
      "authorised_at": null,
      "expires_at": "2021-03-25T17:26:28.305Z",
      "redirect_uri": "https://my-company.com/landing",
      "links": [
        {
          "billing_request": "BRQ123",
          "institution": "monzo"
        }
      ]
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardless\Environment::SANDBOX;
]);
$client->bankAuthorisations()->get("BAU123");
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.bank_authorisations.get("BAU123")
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.bank_authorisations.get("BAU123")
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();
```

```
client.bankAuthorisations().get("BAU123").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.bankAuthorisations.find("BAU123");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);
```

```
var resp = await gocardless.BankAuthorisations.GetAsync("BAU123");
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }
    bankAuthorisation, err := client.BankAuthorisations.Get(context, "BAU123")
}
```

## Billing Requests

Billing Requests help create resources that require input or action from a customer. An example of required input might be additional customer billing details, while an action would be asking a customer to authorise a payment using their mobile banking app.

See [Billing Requests: Overview](#) for how-to's, explanations and tutorials. **Important:** All properties associated with `subscription_request` and `instalment_schedule_request` are only supported for ACH and PAD schemes.

## Properties

## id

Unique identifier, beginning with “BRQ”.

## actions

List of actions that can be performed before this billing request can be fulfilled.

Each instance will contain these properties:

- `available_currencies`
- `bank_authorisation`
- `collect_customer_details`
- `completes_actions`: Value
- `institution_guesses`:
  - `not_needed`: we do not support this feature
  - `pending`: we are processing this request
  - `failed`: we were unable to process this request
  - `success`: we inferred the institution and added it to the resources of a Billing Request

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

if:  
the billing request does not

- **required:** Informs you whether the action is required to fulfil the billing request or not.
- **requires\_actions:** Requires completing these actions before this action can be completed.
- **status:** Status of the action
- **type:** Unique identifier for the action.

**created\_at**

Fixed [timestamp](#), recording when this resource was created.

**fallback\_enabled**

(Optional) If true, this billing request can fallback from instant payment to direct debit. Should not be set if GoCardless payment intelligence feature is used.

See [Billing Requests: Retain customers with Fallbacks](#) for more information.

**fallback\_occurred**

True if the billing request was completed with direct debit.

**instalment\_schedule\_request[app\_fee]**

The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**instalment\_schedule\_request[currency]**

[ISO 4217](#) currency code. Currently “USD” and “CAD” are supported.

**installments\_with\_dates**

An explicit array of instalment payments, each specifying at least an `amount` and `charge_date`. See [create \(with dates\)](#)

Each instance will contain these properties:

- **amount:** Amount, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
- **charge\_date:** A future date on which the payment should be collected. If the date is before the `next_possible_charge_date` on the [mandate](#), it will be automatically rolled forwards to that date.
- **description:** A human-readable description of the payment. This will be included in the notification email GoCardless sends to your customer if your organisation does not send its own notifications (see [compliance requirements](#)).

**amounts**

List of amounts of each instalment, in the lowest denomination for the currency (e.g. cents in USD).

**installments\_with\_schedule[interval]**

Number of `interval_units` between charge dates. Must be greater than or equal to 1.

**installments\_with\_schedule[interval\_unit]**

The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.

**installments\_with\_schedule[start\_date]**

The date on which the first payment should be charged. Must be on or after the [mandate](#)'s `next_possible_charge_date`. When left blank and `month` or `day_of_month` are provided, this will be set to the date of the first payment. If created without `month` or `day_of_month` this will be set as the mandate's `next_possible_charge_date`

**links[instalment\_schedule]**

(Optional) ID of the [instalment\\_schedule](#) that was created from this instalment schedule request.

**instalment\_schedule\_request[metadata]**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**instalment\_schedule\_request[name]**

Name of the instalment schedule, up to 100 chars. This name will also be copied to the payments of the instalment schedule if you use schedule-based creation.

**instalment\_schedule\_request[payment\_reference]**

An optional payment reference. This will be set as the reference on each payment created and will appear on your customer's bank statement. See the documentation for the [create payment endpoint](#) for more details.

**instalment\_schedule\_request[retry\_if\_possible]**

On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

**instalment\_schedule\_request[total\_amount]**

The total amount of the instalment schedule, defined as the sum of all individual payments, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR). If the requested payment amounts do not sum up correctly, a validation error will be returned.

**mandate\_request[authorisation\_source]**

This field is ACH specific, sometimes referred to as [SEC code](#).

This is the way that the payer gives authorisation to the merchant. web: Authorisation is Internet Initiated or via Mobile Entry (maps to SEC code: WEB)  
telephone: Authorisation is provided orally over telephone (maps to SEC code: TEL) paper: Authorisation is provided in writing and signed, or similarly authenticated (maps to SEC code: PPD)

**mandate\_request[consent\_type]**

This attribute represents the authorisation type between the payer and merchant. It can be set to `one_off`, `recurring` or `standing` for ACH scheme. And `single`, `recurring` and `sporadic` for PAD scheme. *Note:* This is only supported for ACH and PAD schemes.

**constraints[end\_date]**

The latest date at which payments can be taken, must occur after `start_date` if present

This is an optional field and if it is not supplied the agreement will be considered open and will not have an end date. Keep in mind the end date must take into account how long it will take the user to set up this agreement via the Billing Request

**constraints[max\_amount\_per\_payment]**

The maximum amount that

**constraints[payment\_method]**

A constraint where you can

**periodic\_limits**

List of periodic limits and c

Each instance will contain t

- **alignment:** The alignm

calendar - this will fi  
year.

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

ACH and PAD schemes.

week or the January for the next

`creation_date` - this will finish on the next instance of the current period. For example Monthly it will expire on the same day of the next month, or yearly the same day of the next year.

- `max_payments`: (Optional) The maximum number of payments that can be collected in this periodic limit.
- `max_total_amount`: The maximum total amount that can be charged for all payments in this periodic limit. Required for VRP.
- `period`: The repeating period for this mandate. Defaults to flexible for PayTo if not specified.

#### constraints[start\_date]

The date from which payments can be taken.

This is an optional field and if it is not supplied the start date will be set to the day authorisation happens.

#### mandate\_request[currency]

[ISO 4217](#) currency code.

#### mandate\_request[description]

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

#### links[mandate]

(Optional) ID of the [mandate](#) that was created from this mandate request.

#### mandate\_request[metadata]

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### mandate\_request[payer\_requested\_dual\_signature]

This attribute can be set to true if the payer has indicated that multiple signatures are required for the mandate. As long as every other Billing Request actions have been completed, the payer will receive an email notification containing instructions on how to complete the additional signature. The dual signature flow can only be completed using GoCardless branded pages.

#### mandate\_request[scheme]

A bank payment scheme. Currently "ach", "autogiro", "bacs", "becs", "becs\_nz", "betalingsservice", "faster\_payments", "pad", "pay\_to" and "sepa\_core" are supported. Optional for mandate only requests - if left blank, the payer will be able to select the currency/scheme to pay with from a list of your available schemes.

#### mandate\_request[sweeping]

If true, this billing request would be used to set up a mandate solely for moving (or sweeping) money from one account owned by the payer to another account that the payer also owns. This is required for Faster Payments

#### mandate\_request[verify]

Verification preference for the mandate. One of:

- `minimum`: only verify if absolutely required, such as when part of scheme rules
- `recommended`: in addition to `minimum`, use the GoCardless payment intelligence solution to decide if a payer should be verified
- `when_available`: if verification mechanisms are available, use them
- `always`: as `when_available`, but fail to create the Billing Request if a mechanism isn't available

By default, all Billing Requests use the recommended verification preference. It uses GoCardless payment intelligence solution to determine if a payer is fraudulent or not. The verification mechanism is based on the response and the payer may be asked to verify themselves. If the feature is not available, `recommended` behaves like `minimum`.

If you never wish to take advantage of our reduced risk products and Verified Mandates as they are released in new schemes, please use the `minimum` verification preference.

See [Billing Requests: Creating Verified Mandates](#) for more information.

#### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### payment\_request[amount]

Amount in minor unit (e.g. pence in GBP, cents in EUR).

#### payment\_request[app\_fee]

The amount to be deducted from the payment as an app fee, to be paid to the partner integration which created the billing request, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

#### payment\_request[currency]

[ISO 4217](#) currency code. GBP and EUR supported; GBP with your customers in the UK and for EUR with your customers in supported Eurozone countries only.

#### payment\_request[description]

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

#### payment\_request[funds\_settlement]

This field will decide how GoCardless handles settlement of funds from the customer.

- `managed` will be moved through GoCardless' account, batched, and payed out.
- `direct` will be a direct transfer from the payer's account to the merchant where invoicing will be handled separately.

#### links[payment]

(Optional) ID of the [payment](#) that was created from this payment request.

#### payment\_request[metadata]

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### payment\_request[reference]

A custom payment reference defined by the merchant. It is only available for payments on the PayTo scheme or payments using the Direct Funds settlement model on the Faster Payments [scheme](#).

#### payment\_request[scheme]

(Optional) A scheme used for

`sepa_instant_credit_trans`

Please be aware that `sepa_i`

#### purpose\_code

Specifies the high-level pur

Currently mortgage, utility,

#### customer[company\_name]

Customer's company name  
that any mandate created fro

#### customer[created\_at]

Fixed [timestamp](#), recording

#### customer[email]

Customer's email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

`edit_transfer` and `transfer` is used as the default.

heme, optional for all others. r are supported.

a `company_name` value will mean idered to be a "Personal PAD").

`customer[family_name]`  
Customer's surname. Required unless a `company_name` is provided.

`customer[given_name]`  
Customer's first name. Required unless a `company_name` is provided.

`customer[id]`  
Unique identifier, beginning with "CU".

`customer[language]`  
[ISO 639-1 code](#). Used as the language for notification emails sent by GoCardless if your organisation does not send its own (see [compliance requirements](#)). Currently only "en", "fr", "de", "pt", "es", "it", "nl", "da", "nb", "sl", "sv" are supported. If this is not provided, the language will be chosen based on the `country_code` (if supplied) or default to "en".

`customer[metadata]`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer[phone_number]`  
[ITU E.123](#) formatted phone number, including country code.

`customer_bank_account[account_holder_name]`  
Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a [customer bank account token](#).

`customer_bank_account[account_number_ending]`  
The last few digits of the account number. Currently 4 digits for NZD bank accounts and 2 digits for other currencies.

`customer_bank_account[account_type]`  
Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.

`customer_bank_account[bank_account_token]`  
A token to uniquely refer to a set of bank account details. This feature is still in early access and is only available for certain organisations.

`customer_bank_account[bank_name]`  
Name of bank, taken from the bank details.

`customer_bank_account[country_code]`  
[ISO 3166-1 alpha-2 code](#). Defaults to the country code of the `iban` if supplied, otherwise is required.

`customer_bank_account[created_at]`  
Fixed [timestamp](#), recording when this resource was created.

`customer_bank_account[currency]`  
[ISO 4217](#) currency code. Currently "AUD", "CAD", "DKK", "EUR", "GBP", "NZD", "SEK" and "USD" are supported.

`customer_bank_account[enabled]`  
Boolean value showing whether the bank account is enabled or disabled.

`customer_bank_account[id]`  
Unique identifier, beginning with "BA".

`links[customer]`  
ID of the [customer](#) that owns this bank account.

`customer_bank_account[metadata]`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer_billing_detail[address_line1]`  
The first line of the customer's address.

`customer_billing_detail[address_line2]`  
The second line of the customer's address.

`customer_billing_detail[address_line3]`  
The third line of the customer's address.

`customer_billing_detail[city]`  
The city of the customer's address.

`customer_billing_detail[country_code]`  
[ISO 3166-1 alpha-2 code](#).

`customer_billing_detail[created_at]`  
Fixed [timestamp](#), recording when this resource was created.

`customer_billing_detail[danish_identity_number]`  
For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer's bank account is denominated in Danish krone (DKK).

`customer_billing_detail[id]`  
Unique identifier, beginning with "CU".

`customer_billing_detail[ip_address]`  
For ACH customers only. Required for ACH customers. A string containing the IP address of the payer to whom the mandate belongs (i.e. as a result of their completion of a mandate setup flow in their browser).

Not required for creating offline mandates where `authorisation_source` is set to telephone or paper.

`customer_billing_detail[postal_code]`  
The customer's postal code.

`customer_billing_detail[region]`  
The customer's address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

`schemes`  
The schemes associated with this customer billing detail

`customer_billing_detail[swedish_identity_number]`  
For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer's bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.

`status`  
One of:

- pending: the billing request has been received but not yet processed.
- ready\_to\_fulfil: the mandate is ready to be fulfilled.
- fulfilling: the billing request is currently being processed.
- fulfilled: the billing request has been successfully processed.
- cancelled: the billing request has been cancelled.

`subscription_request[amount]`  
Amount in the lowest denominator.

`subscription_request[app_fee]`

### Nous utilisons des cookies

- Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

`subscription_request[count]`  
The total number of payments that should be taken by this subscription.

`subscription_request[currency]`  
[ISO 4217](#) currency code. Currently “USD” and “CAD” are supported.

`subscription_request[day_of_month]`  
As per RFC 2445. The day of the month to charge customers on. 1-28 or -1 to indicate the last day of the month.

`subscription_request[interval]`  
Number of interval\_units between customer charge dates. Must be greater than or equal to 1. Must result in at least one charge date per year. Defaults to 1.

`subscription_request[interval_unit]`  
The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.

`links[subscription]`  
(Optional) ID of the [subscription](#) that was created from this subscription request.

`subscription_request[metadata]`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`subscription_request[month]`  
Name of the month on which to charge a customer. Must be lowercase. Only applies when the interval\_unit is `yearly`.

`subscription_request[name]`  
Optional name for the subscription. This will be set as the description on each payment created. Must not exceed 255 characters.

`subscription_request[payment_reference]`  
An optional payment reference. This will be set as the reference on each payment created and will appear on your customer’s bank statement. See the documentation for the [create payment endpoint](#) for more details.

`subscription_request[retry_if_possible]`  
On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

`subscription_request[start_date]`  
The date on which the first payment should be charged. If fulfilled after this date, this will be set as the mandate’s `next_possible_charge_date`. When left blank and month or day\_of\_month are provided, this will be set to the date of the first payment. If created without month or day\_of\_month this will be set as the mandate’s `next_possible_charge_date`.

`links[bank_authorisation]`  
(Optional) ID of the [bank authorisation](#) that was used to verify this request.

`links[creditor]`  
ID of the associated [creditor](#).

`links[customer]`  
ID of the [customer](#) that will be used for this request

`links[customer_bank_account]`  
(Optional) ID of the [customer bank account](#) that will be used for this request

`links[customer_billing_detail]`  
ID of the customer billing detail that will be used for this request

`links[instalment_schedule_request]`  
(Optional) ID of the associated instalment schedule request

`links[instalment_schedule_request_instalment_schedule]`  
(Optional) ID of the [instalment schedule](#) that was created from this instalment schedule request.

`links[mandate_request]`  
(Optional) ID of the associated mandate request

`links[mandate_request_mandate]`  
(Optional) ID of the [mandate](#) that was created from this mandate request. this mandate request.

`links[organisation]`  
ID of the associated organisation.

`links[payment_provider]`  
(Optional) ID of the associated payment provider

`links[payment_request]`  
(Optional) ID of the associated payment request

`links[payment_request_payment]`  
(Optional) ID of the [payment](#) that was created from this payment request.

`links[subscription_request]`  
(Optional) ID of the associated subscription request

`links[subscription_request_subscription]`  
(Optional) ID of the [subscription](#) that was created from this subscription request.

## Create a Billing Request

**Important:** All properties associated with `subscription_request` and `instalment_schedule_request` are only supported for ACH and PAD schemes.

Relative endpoint: POST /billing\_requests

**Note:** Instant Bank Pay is only available for transactions in GBP with your customers in the UK and for EUR with your customers in supported Eurozone countries. We will be adding other countries soon.

**Warning:** by default, the ability to provide a custom mandate reference is switched off. The banking system rules for valid references are quite complex, and we recommend allowing GoCardless to generate it. If you would like to provide custom references, please contact support.

### Parameters

`fallback_enabled`  
(Optional) If true, this billin  
See [Billing Requests: Retail](#)

`instalment_schedule_request[amount]`  
The amount to be deducted  
for the currency (e.g. pence

`instalment_schedule_request[currency]`  
[ISO 4217](#) currency code. C  
instalment\_schedule\_request[in

An explicit array of instalment payments, each specifying at least an amount and charge\_date. See [create \(with dates\)](#).

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

ent intelligence feature is used.

on, in the lowest denomination

**instalment\_schedule\_request[instalments\_with\_schedule]**

Frequency of the payments you want to create, together with an array of payment amounts to be collected, with a specified start date for the first payment. See [create \(with schedule\)](#).

Properties:

- **amounts**: List of amounts of each instalment, in the lowest denomination for the currency (e.g. cents in USD).
- **interval**: Number of **interval\_units** between charge dates. Must be greater than or equal to 1.
- **interval\_unit**: The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.
- **start\_date**: The date on which the first payment should be charged. Must be on or after the [mandate](#)'s `next_possible_charge_date`. When left blank and `month` or `day_of_month` are provided, this will be set to the date of the first payment. If created without `month` or `day_of_month` this will be set as the `mandate's next_possible_charge_date`

**instalment\_schedule\_request[metadata]**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**instalment\_schedule\_request[name]**

Name of the instalment schedule, up to 100 chars. This name will also be copied to the payments of the instalment schedule if you use schedule-based creation.

**instalment\_schedule\_request[payment\_reference]**

An optional payment reference. This will be set as the reference on each payment created and will appear on your customer's bank statement. See the documentation for the [create payment endpoint](#) for more details.

**instalment\_schedule\_request[retry\_if\_possible]**

On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

**instalment\_schedule\_request[total\_amount]**

The total amount of the instalment schedule, defined as the sum of all individual payments, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR). If the requested payment amounts do not sum up correctly, a validation error will be returned.

**mandate\_request[authorisation\_source]**

This field is ACH specific, sometimes referred to as [SEC code](#).

This is the way that the payer gives authorisation to the merchant. web: Authorisation is Internet Initiated or via Mobile Entry (maps to SEC code: WEB) telephone: Authorisation is provided orally over telephone (maps to SEC code: TEL) paper: Authorisation is provided in writing and signed, or similarly authenticated (maps to SEC code: PPD)

**mandate\_request[consent\_type]**

This attribute represents the authorisation type between the payer and merchant. It can be set to `one_off`, `recurring` or `standing` for ACH scheme. And `single`, `recurring` and `sporadic` for PAD scheme. *Note:* This is only supported for ACH and PAD schemes.

**mandate\_request[constraints]**

Constraints that will apply to the mandate\_request. (Optional) Specifically required for PayTo and VRP.

Properties:

- **end\_date**: The latest date at which payments can be taken, must occur after `start_date` if present

This is an optional field and if it is not supplied the agreement will be considered open and will not have an end date. Keep in mind the end date must take into account how long it will take the user to set up this agreement via the Billing Request.

- **max\_amount\_per\_payment**: The maximum amount that can be charged for a single payment. Required for PayTo and VRP.
- **payment\_method**: A constraint where you can specify info (free text string) about how payments are calculated. *Note:* This is only supported for ACH and PAD schemes.
- **periodic\_limits**: List of periodic limits and constraints which apply to them
- **start\_date**: The date from which payments can be taken.

This is an optional field and if it is not supplied the start date will be set to the day authorisation happens.

**mandate\_request[currency]**

[ISO 4217](#) currency code.

**mandate\_request[description]**

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

**mandate\_request[metadata]**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**mandate\_request[reference]**

Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

**mandate\_request[scheme]**

A bank payment scheme. Currently "ach", "autogiro", "bacs", "becs", "becs\_nz", "betalingsservice", "faster\_payments", "pad", "pay\_to" and "sepa\_core" are supported. Optional for mandate only requests - if left blank, the payer will be able to select the currency/scheme to pay with from a list of your available schemes.

**mandate\_request[sweeping]**

If true, this billing request would be used to set up a mandate solely for moving (or sweeping) money from one account owned by the payer to another account that the payer also owns. This is required for Faster Payments

**mandate\_request[verify]**

Verification preference for the mandate. One of:

- **minimum**: only verify if absolutely required, such as when part of schema rules
- **recommended**: in addition to minimum
- **when\_available**: if verification is available
- **always**: as when\_available

By default, all Billing Requests use the `minimum` verification preference. The verification mechanism is determined by the `when_available` setting.

If you never wish to take advantage of the `when_available` verification preference, you can use the `minimum` verification preference.

See [Billing Requests: Creating Verified Mandates](#) for more information.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**payment\_request[amount]**

Amount in minor unit (e.g. pence in GBP, cents in EUR).

**payment\_request[app\_fee]**

The amount to be deducted from the payment as an app fee, to be paid to the partner integration which created the billing request, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**payment\_request[currency]**

[ISO 4217](#) currency code. GBP and EUR supported; GBP with your customers in the UK and for EUR with your customers in supported Eurozone countries only.

**payment\_request[description]**

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

**payment\_request[funds\_settlement]**

This field will decide how GoCardless handles settlement of funds from the customer.

- managed will be moved through GoCardless' account, batched, and payed out.
- direct will be a direct transfer from the payer's account to the merchant where invoicing will be handled separately.

**payment\_request[metadata]**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**payment\_request[reference]**

A custom payment reference defined by the merchant. It is only available for payments on the PayTo scheme or payments using the Direct Funds settlement model on the Faster Payments scheme.

**payment\_request[retry\_if\_possible]**

On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#). **Important:** This is not applicable to IBP and VRP payments.

**payment\_request[scheme]**

(Optional) A scheme used for Open Banking payments. Currently `faster_payments` is supported in the UK (GBP) and `sepa_credit_transfer` and `sepa_instant_credit_transfer` are supported in supported Eurozone countries (EUR). For Eurozone countries, `sepa_credit_transfer` is used as the default.

Please be aware that `sepa_instant_credit_transfer` may incur an additional fee for your customer.

**purpose\_code**

Specifies the high-level purpose of a mandate and/or payment using a set of pre-defined categories. Required for the PayTo scheme, optional for all others.

Currently `mortgage`, `utility`, `loan`, `dependant_support`, `gambling`, `retail`, `salary`, `personal`, `government`, `pension`, `tax` and other are supported.

**subscription\_request[amount]**

Amount in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**subscription\_request[app\_fee]**

The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**subscription\_request[count]**

The total number of payments that should be taken by this subscription.

**subscription\_request[currency]**

[ISO 4217](#) currency code. Currently "AUD", "CAD", "DKK", "EUR", "GBP", "NZD", "SEK" and "USD" are supported.

**subscription\_request[day\_of\_month]**

As per RFC 2445. The day of the month to charge customers on. 1-28 or -1 to indicate the last day of the month.

**subscription\_request[interval]**

Number of `interval_units` between customer charge dates. Must be greater than or equal to 1. Must result in at least one charge date per year. Defaults to 1.

**subscription\_request[interval\_unit]**

The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.

**subscription\_request[metadata]**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**subscription\_request[month]**

Name of the month on which to charge a customer. Must be lowercase. Only applies when the `interval_unit` is `yearly`.

**subscription\_request[name]**

Optional name for the subscription. This will be set as the description on each payment created. Must not exceed 255 characters.

**subscription\_request[payment\_reference]**

An optional payment reference. This will be set as the reference on each payment created and will appear on your customer's bank statement. See the documentation for the [create payment endpoint](#) for more details.

**subscription\_request[retry\_if\_possible]**

On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

**subscription\_request[start\_date]**

The date on which the first payment should be charged. If fulfilled after this date, this will be set as the mandate's `next_possible_charge_date`. When left blank and `month` or `day_of_month` are provided, this will be set to the date of the first payment. If created without `month` or `day_of_month` this will be set as the mandate's `next_possible_charge_date`.

**links[creditor]**

ID of the associated [creditor](#). Only required if your account manages multiple creditors.

**links[customer]**

ID of the [customer](#) against which this request should be made.

**links[customer\_bank\_account]**

(Optional) ID of the [customer bank account](#) against which this request should be made.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/billing_requests HTTP/1.1
```

Content-Type: application/json

{

```
  "billing_requests": {
    "payment_request": {
      "description": "First Pa",
      "amount": "500",
      "currency": "GBP"
    },
    "mandate_request": {
      "scheme": "bacs"
    }
  }
```

}

HTTP/1.1 201 Created

Location: /billing\_requests/BR

Content-Type: application/json

{

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"billing_requests": {
  "id": "BRQ123",
  "created_at": "2021-03-22T12:20:04.397Z",
  "status": "pending",
  "mandate_request": {
    "currency": "GBP",
    "scheme": "bacs",
    "links": {}
  },
  "payment_request": {
    "description": "First Payment",
    "currency": "GBP",
    "amount": 500,
    "scheme": "faster_payments",
    "links": {}
  },
  "metadata": null,
  "links": {
    "customer": "CU123",
    "customer_billing_detail": "CBD123"
  },
  "creditor_name": "Mr Creditor",
  "actions": [
    {
      "type": "choose_currency",
      "required": true,
      "completes_actions": [],
      "requires_actions": [],
      "status": "completed"
    },
    {
      "type": "collect_customer_details",
      "required": true,
      "completes_actions": [],
      "requires_actions": [
        "choose_currency"
      ],
      "status": "pending",
      "collect_customer_details": {
        "incomplete_fields": {
          "customer": [
            "email",
            "given_name",
            "family_name"
          ],
          "customer_billing_detail": [
            "address_line1",
            "city",
            "postal_code",
            "country_code"
          ]
        }
      }
    },
    {
      "type": "select_institution",
      "required": false,
      "completes_actions": [],
      "requires_actions": [],
      "status": "pending"
    },
    {
      "type": "collect_bank_account",
      "required": true,
      "completes_actions": [
        "choose_currency"
      ],
      "requires_actions": [],
      "status": "pending"
    },
    {
      "type": "bank_authorisation",
      "required": true,
      "completes_actions": [],
      "requires_actions": [
        "collect_bank_account"
      ],
      "status": "pending"
    }
  ],
  "resources": {
    "customer": {
      "id": "CU123",
      "created_at": "2021-03-22T12:20:04.238Z",
      "email": null,
      "given_name": null,
      "family_name": null,
      "company_name": null,
      "language": "en",
      "phone_number": null,
      "metadata": {}
    },
    "customer_billing_": {
      "id": "CBD123",
      "created_at": "2021-03-22T12:20:04.238Z",
      "address_line1": null,
      "address_line2": null,
      "address_line3": null,
      "city": null,
      "region": null,
      "postal_code": null,
      "country_code": null,
      "swedish_identity": null,
      "danish_identity_number": null
    }
  }
}

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        }
    }
}
PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequests()->create([
    "params" => [
        "payment_request" => [
            "description" => "First Payment",
            "amount"      => "500",
            "currency"    => "GBP",
        ],
        "mandate_request" => [
            "scheme" => "bacs"
        ]
    ]
]);

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.create(params={
    "payment_request": {
        "description": "First Payment",
        "amount": "500",
        "currency": "GBP",
    },
    "mandate_request": {
        "scheme": "bacs"
    }
})

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_requests.create(
  params: {
    payment_request: {
      description: "First Payment",
      amount: "500",
      currency: "GBP",
    },
    mandate_request: {
      scheme: "bacs"
    }
  }
)

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

BillingRequest billingRequest = client.billingRequests().create()
    .withPaymentRequestDescription("First Payment")
    .withPaymentRequestAmount(500)
    .withPaymentRequestCurrency(Currency.GBP)
    .withMandateRequestScheme("bacs")
    .execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const billingRequest = await client.billingRequests.create({
    payment_request: {
        description: "First Payment",
        amount: "500",
        currency: "GBP",
    },
    mandate_request: {
        scheme: "bacs"
    }
});

```

## .NET

```

String accessToken = "your_acco
GoCardlessClient gocardless = (
    var paymentRequest = new gocar
    {
        Description = "First payment
        Amount = 500,
        Currency = "GBP",
    };

```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var mandateRequest = new gocardless.BillingRequestCreateRequest.BillingRequestMandateRequest
{
  Scheme = "bacs",
};

var resp = await gocardless.BillingRequests.CreateAsync(
  new GoCardless.Services.BillingRequestCreateRequest()
  {
    PaymentRequest = paymentRequest,
    MandateRequest = mandateRequest,
  }
);

GoCardless.Resources.BillingRequest billingRequest = resp.BillingRequest;
Go

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  billingRequestCreateParams := gocardless.BillingRequestCreateParams{
    PaymentRequest: &gocardless.BillingRequestCreateParamsPaymentRequest{
      Amount:        1000,
      Currency:     "GBP",
      Description:  "First Payment",
    },
    MandateRequest: &gocardless.BillingRequestCreateParamsMandateRequest{
      Scheme: "bacs",
    },
  }

  billingRequest, err := client.BillingRequests.Create(context, billingRequestCreateParams)
}

```

## Collect customer details

If the billing request has a pending `collect_customer_details` action, this endpoint can be used to collect the details in order to complete it.

The endpoint takes the same payload as Customers, but checks that the customer fields are populated correctly for the billing request scheme.

Whatever is provided to this endpoint is used to update the referenced customer, and will take effect immediately after the request is successful.

Relative endpoint: POST `/billing_requests/BRQ123/actions/collect_customer_details`

**Restricted:** this endpoint is restricted to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with the [custom payment pages upgrade](#).

### Parameters

`customer[company_name]`

Customer's company name. Required unless a `given_name` and `family_name` are provided. For Canadian customers, the use of a `company_name` value will mean that any mandate created from this customer will be considered to be a "Business PAD" (otherwise, any mandate will be considered to be a "Personal PAD").

`customer[email]`

Customer's email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

`customer[family_name]`

Customer's surname. Required unless a `company_name` is provided.

`customer[given_name]`

Customer's first name. Required unless a `company_name` is provided.

`customer[language]`

[ISO 639-1](#) code. Used as the language for notification emails sent by GoCardless if your organisation does not send its own (see [compliance requirements](#)). Currently only "en", "fr", "de", "pt", "es", "it", "nl", "da", "nb", "sl", "sv" are supported. If this is not provided and a customer was linked during billing request creation, the linked customer language will be used. Otherwise, the language is default to "en".

`customer[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer[phone_number]`

[ITU E.123](#) formatted phone number including country code.

`customer_billing_detail[address]`

The first line of the customer's address.

`customer_billing_detail[address_2]`

The second line of the customer's address.

`customer_billing_detail[address_3]`

The third line of the customer's address.

`customer_billing_detail[city]`

The city of the customer's address.

`customer_billing_detail[country]`

[ISO 3166-1 alpha-2 code](#).

`customer_billing_detail[danish_krone]`

For Danish customers only.

Danish krone (DKK).

`customer_billing_detail[ip_address]`

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

nk account is denominated in

For ACH customers only. Required for ACH customers. A string containing the IP address of the payer to whom the mandate belongs (i.e. as a result of their completion of a mandate setup flow in their browser).

Not required for creating offline mandates where `authorisation_source` is set to telephone or paper.

```
customer_billing_detail[postal_code]
The customer's postal code.
customer_billing_detail[region]
The customer's address region, county or department. For US customers a 2 letter ISO3166-2:US state code is required (e.g. CA for California).
customer_billing_detail[swedish_identity_number]
For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer's bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.
```



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST [https://api.gocardless.com/billing\\_requests/BRQ123/actions/collect\\_customer\\_details](https://api.gocardless.com/billing_requests/BRQ123/actions/collect_customer_details) HTTP/1.1

Content-Type: application/json

```
{
  "data": {
    "customer": {
      "email": "alice@example.com",
      "given_name": "Alice",
      "family_name": "Smith"
    },
    "customer_billing_detail": {
      "address_line1": "1 Somewhere Lane"
    }
  }
}
```

HTTP/1.1 200

Content-Type: application/json

```
{
  "billing_request": {
    "id": "BRQ123",
    "created_at": "2021-03-22T12:20:04.397Z",
    "status": "pending",
    "mandate_request": {
      "currency": "GBP",
      "scheme": "bacs",
      "links": {}
    },
    "payment_request": {
      "description": "First Payment",
      "currency": "GBP",
      "amount": 500,
      "scheme": "faster_payments",
      "links": {}
    },
    "metadata": null,
    "links": {
      "customer": "CU123",
      "creditor": "CR123",
      "mandate_request": "MRQ123"
    },
    "creditor_name": "Mr Creditor",
    "actions": [
      {
        "type": "choose_currency",
        "required": true,
        "completes_actions": [],
        "requires_actions": [],
        "status": "completed"
      },
      {
        "type": "collect_customer_details",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "choose_currency"
        ],
        "status": "pending",
        "collect_customer_details": {
          "incomplete_fields": {
            "customer": [],
            "customer_billing_detail": [
              "address_line1",
              "city",
              "postal_code",
              "country_code"
            ]
          }
        }
      },
      {
        "type": "select_currency",
        "required": false,
        "completes_actions": [],
        "requires_actions": [],
        "status": "pending"
      },
      {
        "type": "collect_customer_details",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "choose_currency"
        ],
        "status": "pending"
      }
    ]
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "type": "bank_authorisation",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
            "collect_bank_account"
        ],
        "status": "pending"
    },
    "resources": {
        "customer": {
            "id": "CU123",
            "created_at": "2021-03-22T12:20:04.238Z",
            "email": "alice@example.com",
            "given_name": "Alice",
            "family_name": "Smith",
            "company_name": null,
            "language": "en",
            "phone_number": null,
            "metadata": {}
        }
    }
}
}
PHP

```

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequests()->collectCustomerDetails("BR123", [
    "params" => [
        "customer" => [
            "email" => "alice@example.com",
            "given_name" => "Alice",
            "family_name" => "Smith"
        ],
        "customer_billing_detail" => [
            "address_line1" => "1 Somewhere Lane"
        ]
    ]
]);

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.collect_customer_details("BR123", params={
    customer: {
        email: "alice@example.com",
        given_name: "Alice",
        family_name: "Smith",
    },
    customer_billing_detail: {
        address_line1: "1 Somewhere Lane"
    }
})

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_requests.collect_customer_details("BR123", {
  params: {
    customer: {
      email: "alice@example.com",
      given_name: "Alice",
      family_name: "Smith",
    },
    customer_billing_detail: {
      address_line1: "1 Somewhere Lane"
    }
  }
})

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.billingRequests().collectCustomerDetails()
    .withCustomerEmail("alice@example.com")
    .withCustomerGivenName("Alice")
    .withCustomerFamilyName("Smith")
    .withCustomerBillingDetailAddressLine1("1 Somewhere Lane")
    .execute();

```

## JavaScript

```

const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const resp = await client.billingRequests().collectCustomerDetails()
    .withCustomerEmail("alice@example.com")
    .withCustomerGivenName("Alice")
    .withCustomerFamilyName("Smith")
    .withCustomerBillingDetailAddressLine1("1 Somewhere Lane")
    .execute();

```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

email: "alice@example.com",
given_name: "Alice",
family_name: "Smith",
},
customer_billing_detail: {
  address_line1: "1 Somewhere Lane"
}
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customer = new gocardless.BillingRequests.CollectCustomerDetailsRequest.BillingRequestCustomer
{
  Email = "alice@example.com",
  GivenName = "Alice",
  FamilyName = "Smith",
};

var customerBillingDetail = new gocardless.BillingRequests.CollectCustomerDetailsRequest.BillingRequestCustomerBillingDetail
{
  AddressLine1 = "1 Somewhere Lane",
};

var resp = await gocardless.BillingRequests.CollectCustomerDetails("BR123",
  new BillingRequestCollectCustomerDetailsRequest
  {
    Customer = customer,
    CustomerBillingDetail = customerBillingDetail,
  });
}

Go

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  billingRequestCollectCustomerDetailsParams := gocardless.BillingRequestCollectCustomerDetailsParams{
    Customer: &gocardless.BillingRequestCollectCustomerDetailsParamsCustomer{
      GivenName: "Alice",
      FamilyName: "Smith",
      Email: "alice@example.com",
    },
    CustomerBillingDetail: &gocardless.BillingRequestCollectCustomerDetailsParamsCustomerBillingDetail{
      AddressLine1: "1 Somewhere Lane",
    },
  }

  billingRequest, err := client.BillingRequests.CollectCustomerDetails(context, "BR123", billingRequestCollectCustomerDetailsParams)
}

```

## Collect bank account details

If the billing request has a pending `collect_bank_account` action, this endpoint can be used to collect the details in order to complete it.

The endpoint takes the same payload as Customer Bank Accounts, but check the bank account is valid for the billing request scheme before creating and attaching it.

If the scheme is PayTo and the `pay_id` is available, this can be included in the payload along with the `country_code`.

*ACH scheme* For compliance reasons, an extra validation step is done using a third-party provider to make sure the customer's bank account can accept Direct Debit. If a bank account is discovered to be closed or invalid, the customer is requested to adjust the account number/routing number and succeed in this check to continue with the flow.

*BACS scheme* [Payer Name Verification](#) is enabled by default for UK based bank accounts, meaning we verify the account holder name and bank account number match the details held by the relevant bank.

Relative endpoint: POST `/billing_requests/{REQUEST_ID}/actions/collect_bank_account`

**Restricted:** this endpoint is restricted to users with the [Banker](#) role.

Parameters

`account_holder_name`  
Name of the account holder  
request includes a [customer](#)  
`account_number`  
Bank account number - see [local details](#)  
`account_number_suffix`  
Account number suffix (only)  
`account_type`  
Bank account type. Required information.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

[Lire](#)

This field is required unless the

...es. See [local details](#) for more

**bank\_code**  
Bank code - see [local details](#) for more information. Alternatively you can provide an iban.

**branch\_code**  
Branch code - see [local details](#) for more information. Alternatively you can provide an iban.

**country\_code**  
[ISO 3166-1 alpha-2 code](#). Defaults to the country code of the iban if supplied, otherwise is required.

**currency**  
[ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

**iban**  
International Bank Account Number. Alternatively you can provide [local details](#). IBANs are not accepted for Swedish bank accounts denominated in SEK - you must supply [local details](#).

**metadata**  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**pay\_id**  
A unique record such as an email address, mobile number or company number, that can be used to make and accept payments.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/billing_requests/BRQ123/actions/collect_bank_account HTTP/1.1
Content-Type: application/json
{
  "data": {
    "account_number": "55779911",
    "branch_code": "200000",
    "account_holder_name": "Frank Osborne",
    "country_code": "GB"
  }
}

HTTP/1.1 200
Content-Type: application/json
{
  "billing_request": {
    "id": "BRQ123",
    "created_at": "2021-03-22T12:20:04.397Z",
    "status": "pending",
    "mandate_request": {
      "currency": "GBP",
      "scheme": "bacs",
      "links": {}
    },
    "payment_request": {
      "description": "First Payment",
      "currency": "GBP",
      "amount": 500,
      "scheme": "faster_payments",
      "links": {}
    },
    "metadata": null,
    "links": {
      "customer": "CU123",
      "customer_bank_account": "BA123",
      "creditor": "CR123",
      "mandate_request": "MRQ123"
    },
    "creditor_name": "Mr Creditor",
    "actions": [
      {
        "type": "choose_currency",
        "required": true,
        "completes_actions": [],
        "requires_actions": [],
        "status": "completed"
      },
      {
        "type": "collect_customer_details",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "choose_currency"
        ],
        "status": "pending",
        "collect_customer_details": {
          "incomplete_fields": [
            "customer": [
              "email",
              "given_name",
              "family_name"
            ],
            "customer_billing_detail": [
              "address_line1",
              "city",
              "post_code",
              "country"
            ]
          ]
        }
      }
    ]
  }
}

{
  "type": "select_currency",
  "required": false,
  "completes_actions": [],
  "requires_actions": [],
  "status": "pending"
},
{
  "type": "collect_customer_details",
  "required": true,
  "completes_actions": [
    "choose_currency"
  ],
  "status": "pending"
}

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

```

        ],
        "requires_actions": [],
        "status": "pending"
    },
    {
        "type": "bank_authorisation",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
            "collect_bank_account"
        ],
        "status": "pending"
    }
],
"resources": {
    "customer": {
        "id": "CU123",
        "created_at": "2021-03-22T12:20:04.238Z",
        "email": null,
        "given_name": "Alice",
        "family_name": "Smith",
        "company_name": null,
        "language": "en",
        "phone_number": null,
        "metadata": {}
    },
    "customer_bank_account": {
        "id": "BA123",
        "created_at": "2014-05-08T17:01:06.000Z",
        "account_holder_name": "Frank Osborne",
        "account_numberEnding": "11",
        "country_code": "GB",
        "currency": "GBP",
        "bank_name": "BARCLAYS BANK PLC",
        "metadata": {},
        "enabled": true,
        "links": {
            "customer": "CU123"
        }
    }
}
}
}
PHP

```

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequests()->collectBankAccount("BR123", [
    "params" => [
        "account_number" => "55779911",
        "branch_code" => "200000",
        "account_holder_name" => "Frank Osborne",
        "country_code" => "GB"
    ]
]);

```

#### Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.collect_bank_account("BR123", params={
    account_number: "55779911",
    branch_code: "200000",
    account_holder_name: "Frank Osborne",
    country_code: "GB",
})

```

#### Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_requests.collect_bank_account("BR123", {
  params: {
    account_number: "55779911",
    branch_code: "200000",
    account_holder_name: "Frank Osborne",
    country_code: "GB",
  }
})

```

#### Java

```

import static com.gocardless.G
String accessToken = "your_acce
GoCardlessClient client = GoCa
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.billingRequests().collect
    .withAccountNumber("55779911")
    .withBranchCode("200000")
    .withAccountHolderName("Fran
    .withCountryCode("GB")
    .execute();

```

#### JavaScript

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.billingRequests.collectBankAccount("BR123", {
  account_number: "55779911",
  branch_code: "200000",
  account_holder_name: "Frank Osborne",
  country_code: "GB",
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequests.CollectBankAccountAsync("BR123",
  new BillingRequestCollectBankAccountRequest
  {
    AccountNumber = "55779911",
    BranchCode = "200000",
    AccountHolderName = "Frank Osborne",
    CountryCode = "GB",
  });

```

Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  billingRequestCollectBankAccountParams := gocardless.BillingRequestCollectBankAccountParams{
    BranchCode:      "200000",
    CountryCode:     "GB",
    AccountNumber:   "55779911",
    AccountHolderName: "Frank Osborne",
  }

  billingRequest, err := client.BillingRequests.CollectBankAccount(context, "BR123", billingRequestCollectBankAccountParams)
}

```

## Confirm the payer details

This is needed when you have a mandate request. As a scheme compliance rule we are required to allow the payer to crosscheck the details entered by them and confirm it.

Relative endpoint: POST /billing\_requests/BRQ123/actions/confirm\_payer\_details

Parameters

`metadata`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`payer_requested_dual_signature`

This attribute can be set to true if the payer has indicated that multiple signatures are required for the mandate. As long as every other Billing Request actions have been completed, the payer will receive an email notification containing instructions on how to complete the additional signature. The dual signature flow can only be completed using GoCardless branded pages.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST [https://api.gocardless.com/billing\\_requests/BRQ123/actions/confirm\\_payer\\_details](https://api.gocardless.com/billing_requests/BRQ123/actions/confirm_payer_details) HTTP/1.1

HTTP/1.1 200

Content-Type: application/json

```
{
  "billing_requests": {
    "id": "BRQ123",
    "created_at": "2021-03-15T12:00:00Z",
    "status": "ready_to_fund",
    "mandate_request": {
      "currency": "GBP",
      "scheme": "bacs",
      "links": {}
    },
    "payment_request": null,
    "metadata": null,
    "links": {
      "customer": "CU123",
      "customer_bank_account": "CB123",
      "creditor": "CR123",
      "mandate_request": null
    },
    "creditor_name": "Mr Client",
    "actions": []
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    },
    {
      "type": "choose_currency",
      "required": true,
      "completes_actions": [],
      "requires_actions": [],
      "status": "completed"
    },
    {
      "type": "collect_customer_details",
      "required": true,
      "completes_actions": [],
      "requires_actions": [
        "choose_currency"
      ],
      "status": "completed"
    },
    {
      "type": "collect_bank_account",
      "required": true,
      "completes_actions": [
        "choose_currency"
      ],
      "requires_actions": [],
      "status": "completed"
    },
    {
      "type": "confirm_payer_details",
      "required": true,
      "completes_actions": [],
      "requires_actions": [
        "collect_customer_details",
        "collect_bank_account"
      ],
      "status": "completed"
    }
  ],
  "resources": {
    "customer": {
      "id": "CU123",
      "created_at": "2021-03-22T12:20:04.238Z",
      "email": null,
      "given_name": "Alice",
      "family_name": "Smith",
      "company_name": null,
      "language": "en",
      "phone_number": null,
      "metadata": {}
    },
    "customer_bank_account": {
      "id": "BA123",
      "created_at": "2022-11-01T16:30:37.969Z",
      "account_numberEnding": "11",
      "account_holder_name": "HOME LOAN ACCOUNT",
      "account_type": null,
      "bank_name": "BARCLAYS BANK PLC",
      "currency": "GBP",
      "country_code": "GB",
      "metadata": {},
      "enabled": true,
      "links": {
        "customer": "CU123"
      },
      "branch_codeEnding": "00"
    }
  }
}
}
PHP

```

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->billingRequests()->confirmPayerDetails("BR123");
```

#### Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.confirm_payer_details("BR123")
```

#### Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.billing_requests.confir
```

#### Java

```
import static com.gocardless.G
String accessToken = "your_acce
GoCardlessClient client = GoCa
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
client.billingRequests().confir
```

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.billingRequests.confirmPayerDetails("BR123");
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequests.ConfirmPayerDetailsAsync("BR123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    BillingRequestConfirmPayerDetailsParams := gocardless.BillingRequestConfirmPayerDetailsParams{}
    billingRequest, err := client.BillingRequests.ConfirmPayerDetails(context, "BR123", BillingRequestConfirmPayerDetailsParams)
}

```

## Fulfil a Billing Request

If a billing request is ready to be fulfilled, call this endpoint to cause it to fulfil, executing the payment.

Relative endpoint: POST /billing\_requests/BRQ123/actions/fulfil

Parameters

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com/billing\_requests/BRQ123/actions/fulfil HTTP/1.1

HTTP/1.1 200

Content-Type: application/json

```
{
    "billing_request": {
        "id": "BRQ123",
        "created_at": "2021-03-22T12:20:04.397Z",
        "status": "fulfilled",
        "mandate_request": {
            "currency": "GBP",
            "scheme": "bacs",
            "links": {
                "mandate": "MD123"
            }
        },
        "payment_request": {
            "description": "First Payment",
            "currency": "GBP",
            "amount": 500,
            "scheme": "faster_payments",
            "links": {
                "payment": "PM123"
            }
        },
        "metadata": null,
        "links": {
            "customer": "CU123",
            "customer_billing_detail": "CBD123"
        },
        "creditor_name": "Mr Creditor",
        "actions": [],
        "resources": {
            "customer": {
                "id": "CU123",
                "created_at": null,
                "email": null,
                "given_name": null,
                "family_name": null,
                "company_name": null,
                "language": "en",
                "phone_number": null,
                "metadata": {}
            },
            "customer_billing_": {
                "id": "CBD123",
                "created_at": null,
                "address_line1": "1 Somewhene Lane",
                "address_line2": null,
                "city": null,
                "state": null,
                "zip": null
            }
        }
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "address_line2": null,
        "address_line3": null,
        "city": null,
        "region": null,
        "postal_code": null,
        "country_code": null,
        "swedish_identity_number": null,
        "danish_identity_number": null
    }
}
}

PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequests()->fulfil("BR123");

Python

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.fulfil("BR123")

Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_requests.fulfil("BR123")

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.billingRequests().fulfil("BR123").execute();

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.billingRequests.fulfil("BR123");

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequests.FulfilAsync("BR123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestFulfilParams := gocardless.BillingRequestFulfilParams{}
    billingRequest, err := client.BillingRequests.Fulfil(context, "BR123", billingRequestFulfilParams)
}
}
```

### Cancel a Billing Request

Immediately cancels a billing request.

Relative endpoint: POST /billing

Parameters

metadata

Key-value store of custom data.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)



HTTP PHP Python Ruby Java JavaScript .NET Go

rs.

## HTTP

```
POST https://api.gocardless.com/billing_requests/BRQ123/actions/cancel HTTP/1.1
```

```
HTTP/1.1 200
Content-Type: application/json
{
  "billing_request": {
    "id": "BRQ123",
    "created_at": "2021-03-22T12:20:04.397Z",
    "status": "cancelled",
    "mandate_request": {
      "currency": "GBP",
      "scheme": "bacs",
      "links": {}
    },
    "payment_request": {
      "description": "First Payment",
      "currency": "GBP",
      "amount": 500,
      "scheme": "faster_payments",
      "links": {}
    },
    "metadata": null,
    "links": {
      "customer": "CU123",
      "customer_billing_detail": "CBD123"
    },
    "creditor_name": "Mr Creditor",
    "actions": [],
    "resources": {
      "customer": {
        "id": "CU123",
        "created_at": "2021-03-22T12:20:04.238Z",
        "email": null,
        "given_name": "Alice",
        "family_name": "Smith",
        "company_name": null,
        "language": "en",
        "phone_number": null,
        "metadata": {}
      },
      "customer_billing_detail": {
        "id": "CBD123",
        "created_at": "2021-03-22T12:20:04.374Z",
        "address_line1": "1 Somewhere Lane",
        "address_line2": null,
        "address_line3": null,
        "city": null,
        "region": null,
        "postal_code": null,
        "country_code": null,
        "swedish_identity_number": null,
        "danish_identity_number": null
      }
    }
  }
}
```

## PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->billingRequests()->cancel("BR123");
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.cancel("BR123")
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.billing_requests.cancel("BR123")
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_acce
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
client.billingRequests().cancel("BR123")
```

## JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_acce
const resp = await client.billingRequ
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequests.CancelAsync("BR123");
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestCancelParams := gocardless.BillingRequestCancelParams{}
    billingRequest, err := client.BillingRequests.Fulfil(context, "BR123", billingRequestCancelParams)
}

```

## List Billing Requests

Returns a [cursor-paginated](#) list of your billing requests.

Relative endpoint: GET /billing\_requests

### Parameters

**after** Cursor pointing to the start of the desired set.

**before** Cursor pointing to the end of the desired set.

**created\_at** Fixed [timestamp](#), recording when this resource was created.

**customer** ID of a [customer](#). If specified, this endpoint will return all requests for the given customer.

**limit** Number of records to return.

**status** One of:

- pending: the billing request is pending and can be used
- ready\_to\_fulfil: the billing request is ready to fulfil
- fulfilling: the billing request is currently undergoing fulfilment
- fulfilled: the billing request has been fulfilled and a payment created
- cancelled: the billing request has been cancelled and cannot be used

🕒 ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/billing\_requests HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

```
{
    "meta": {
        "cursors": {
            "before": null,
            "after": null
        },
        "limit": 50
    },
    "billing_requests": [
        {
            "id": "BRQ123",
            "created_at": "2021-03-27T16:00:00Z",
            "status": "pending",
            "mandate_request": null,
            "payment_request": {
                "description": "£5",
                "currency": "GBP",
                "amount": 500,
                "scheme": "faster_payment",
                "links": {}
            },
            "metadata": null,
            "links": {
                "customer": "CU123",
                "customer_billing_details": null,
                "customer_bank_account": null
            }
        }
    ]
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequests()->list();
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.list().records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.billing_requests.list
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

for (BillingRequest billingRequest : client.billingRequests().all().execute()) {
    System.out.println(billingRequest.getId());
}
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const billingRequests = await client.billingRequests.list();
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var billingRequestListResponse = gocardless.BillingRequests.All();
foreach (GoCardless.Resources.BillingRequest billingRequest in billingRequestListResponse)
{
    Console.WriteLine(billingRequest.Id);
}
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestListParams := gocardless.BillingRequestListParams{}
    billingRequestListResult, err := client.BillingRequests.List(context, billingRequestListParams)
    for _, billingRequest := range billingRequestListResult.BillingRequests {
        fmt.Println(billingRequest.Id)
    }
}
```

## Get a single Billing Request

Fetches a billing request

Relative endpoint: GET /billing\_requests/{id}

○ ○ ○ ○ ○ ○ ○ ○  
HTTP PHP Python Ruby Java Java  
HTTP

GET https://api.gocardless.com/billing\_requests/1234567890  
HTTP/1.1 200  
Content-Type: application/json

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
{
  "billing_requests": {
    "id": "BRQ123",
    "created_at": "2021-03-22T12:20:04.397Z",
    "status": "pending",
    "mandate_request": {
      "currency": "GBP",
      "scheme": "bacs",
      "links": {}
    },
    "payment_request": {
      "description": "First Payment",
      "currency": "GBP",
      "amount": 500,
      "scheme": "faster_payments",
      "links": {}
    },
    "metadata": null,
    "links": {
      "customer": "CU123",
      "customer_billing_detail": "CBD123"
    },
    "creditor_name": "Mr Creditor",
    "actions": [
      {
        "type": "choose_currency",
        "required": true,
        "completes_actions": [],
        "requires_actions": [],
        "status": "completed"
      },
      {
        "type": "collect_customer_details",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "choose_currency"
        ],
        "status": "pending",
        "collect_customer_details": {
          "incomplete_fields": {
            "customer": [
              "email",
              "given_name",
              "family_name"
            ],
            "customer_billing_detail": [
              "address_line1",
              "city",
              "postal_code",
              "country_code"
            ]
          }
        }
      },
      {
        "type": "select_institution",
        "required": false,
        "completes_actions": [],
        "requires_actions": [],
        "status": "pending"
      },
      {
        "type": "collect_bank_account",
        "required": true,
        "completes_actions": [
          "choose_currency"
        ],
        "requires_actions": [],
        "status": "pending"
      },
      {
        "type": "bank_authorisation",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "collect_bank_account"
        ],
        "status": "pending"
      }
    ],
    "resources": {
      "customer": {
        "id": "CU123",
        "created_at": "2021-03-22T12:20:04.238Z",
        "email": null,
        "given_name": null,
        "family_name": null,
        "company_name": null,
        "language": "en",
        "phone_number": null,
        "metadata": {}
      },
      "customer_billing": {
        "id": "CBD123",
        "created_at": null,
        "address_line1": null,
        "address_line2": null,
        "address_line3": null,
        "city": null,
        "region": null,
        "postal_code": null,
        "country_code": null,
        "swedish_identity_number": null,
        "links": {}
      }
    }
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "danish_identity_number": null
    }
}
}

PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequests()->get("BR123");

Python

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.get("BR123")

Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_requests.get("BR123")

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.billingRequests().get("BR123").execute();

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.billingRequests.find("BR123");

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequests.GetAsync("BR123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }
    billingRequest, err := client.BillingRequests.Get(context, "BR123")
}

```

## Notify the customer

Notifies the customer linked to the billing request, asking them to authorise it. Currently, the customer can only be notified by email.

This endpoint is currently supported only for Instant Bank Pay Billing Requests

Relative endpoint: POST /billing

Parameters

`notification_type`  
required Currently, can only  
`redirect_uri`  
URL that the payer can be re



HTTP PHP Python Ruby Java Java  
HTTP

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
POST https://api.gocardless.com/billing_requests/BRQ123/actions/notify HTTP/1.1
```

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

```
{
  "data": {
    "notification_type": "email",
    "redirect_uri": "https://my-company.com"
  }
}

HTTP/1.1 200
Content-Type: application/json
{
  "billing_requests": {
    "id": "BRQ123",
    "created_at": "2021-03-22T12:20:04.397Z",
    "status": "pending",
    "mandate_request": {
      "currency": "GBP",
      "scheme": "bacs",
      "links": {}
    },
    "payment_request": {
      "description": "First Payment",
      "currency": "GBP",
      "amount": 500,
      "scheme": "faster_payments",
      "links": {}
    },
    "metadata": null,
    "links": {
      "customer": "CU123",
      "customer_billing_detail": "CBD123"
    },
    "creditor_name": "Mr Creditor",
    "actions": [
      {
        "type": "choose_currency",
        "required": true,
        "completes_actions": [],
        "requires_actions": [],
        "status": "completed"
      },
      {
        "type": "collect_customer_details",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "choose_currency"
        ],
        "status": "pending",
        "collect_customer_details": {
          "incomplete_fields": {
            "customer": [
              "email",
              "given_name",
              "family_name"
            ],
            "customer_billing_detail": [
              "address_line1",
              "city",
              "postal_code",
              "country_code"
            ]
          }
        }
      },
      {
        "type": "select_institution",
        "required": false,
        "completes_actions": [],
        "requires_actions": [],
        "status": "pending"
      },
      {
        "type": "collect_bank_account",
        "required": true,
        "completes_actions": [
          "choose_currency"
        ],
        "requires_actions": [],
        "status": "pending"
      },
      {
        "type": "bank_authorisation",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "collect_bank_account"
        ],
        "status": "pending"
      }
    ],
    "resources": {
      "customer": {
        "id": "CU123",
        "created_at": null,
        "email": null,
        "given_name": null,
        "family_name": null,
        "company_name": null,
        "language": "en",
        "phone_number": null,
        "metadata": {}
      }
    }
  }
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "customer_billing_detail": {
            "id": "CBD123",
            "created_at": "2021-03-22T12:20:04.374Z",
            "address_line1": "1 Somewhere Lane",
            "address_line2": null,
            "address_line3": null,
            "city": null,
            "region": null,
            "postal_code": null,
            "country_code": null,
            "swedish_identity_number": null,
            "danish_identity_number": null
        }
    }
}

```

PHP

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequests()->notify("BR123", [
    'params' => [
        'notification_type' => "email",
        'redirect_uri'      => "https://my-company.com"
    ]
]);

```

Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_requests.notify("BR123", params={
    notification_type: "email",
    redirect_uri: "https://my-company.com",
})

```

Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_requests.notify("BR123", {
  params: {
    notification_type: "email",
    redirect_uri: "https://my-company.com",
  }
})

```

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.billingRequests().notify("BR123")
    .withNotificationType("email")
    .withRedirectUri("https://my-company.com")
    .execute();

```

JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.billingRequests.notify("BR123", {
    notification_type: "email",
    redirect_uri: "https://my-company.com",
});

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequests.NotifyAsync("BR123",
    new BillingRequestNotifyRequest
    {
        NotificationType = "email",
        RedirectUri = "https://my-");
    });

```

Go

```

package main

import (
    gocardless "github.com/gocardless/gocardless"
)

func main() {
    accessToken := "your_access_token"
    config, err := gocardless.NewConfig()
    if err != nil {
        fmt.Printf("got err in initialising config. %s , %v\n", err.Error())
        return
    }
}

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

}
client, err := gocardless.New(config)
if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
}

billingRequestNotifyParams := gocardless.BillingRequestNotifyParams{
    NotificationType: "email",
    RedirectUri:       "https://my-company.com",
}
billingRequest, err := client.BillingRequests.Fulfil(context, "BR123", billingRequestNotifyParams)
}

```

## Trigger fallback

Triggers a fallback from the open-banking flow to direct debit. Note, the billing request must have fallback enabled.

Relative endpoint: POST /billing\_requests/BRQ123/actions/fallback

**Restricted:** this endpoint is restricted to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with the [custom payment pages upgrade](#).

Parameters



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST [https://api.gocardless.com/billing\\_requests/BRQ123/actions/fallback](https://api.gocardless.com/billing_requests/BRQ123/actions/fallback) HTTP/1.1

```

HTTP/1.1 200
Content-Type: application/json
{
    "billing_requests": {
        "id": "BRQ123",
        "created_at": "2021-03-22T12:20:04.397Z",
        "status": "pending",
        "mandate_request": {
            "currency": "GBP",
            "scheme": "bacs",
            "links": {}
        },
        "payment_request": {
            "description": "First Payment",
            "currency": "GBP",
            "amount": 500,
            "scheme": "bacs",
            "links": {}
        },
        "metadata": null,
        "links": {
            "customer": "CU123",
            "customer_billing_detail": "CBD123"
        },
        "creditor_name": "Mr Creditor",
        "actions": [
            {
                "type": "choose_currency",
                "required": true,
                "completes_actions": [],
                "requires_actions": [],
                "status": "completed"
            },
            {
                "type": "collect_customer_details",
                "required": true,
                "completes_actions": [],
                "requires_actions": [
                    "choose_currency"
                ],
                "status": "pending",
                "collect_customer_details": {
                    "incomplete_fields": {
                        "customer": [
                            "email",
                            "given_name",
                            "family_name"
                        ],
                        "customer_billing_detail": [
                            "address_line1",
                            "city",
                            "postal_code",
                            "cc"
                        ]
                    }
                }
            },
            {
                "type": "select_cc",
                "required": false,
                "completes_actions": [],
                "requires_actions": [
                    "choose_currency"
                ],
                "status": "completed"
            },
            {
                "type": "collect_cc",
                "required": true,
                "completes_actions": [
                    "choose_currency"
                ],
                "status": "completed"
            }
        ]
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Consultez notre politique de confidentialité

```

        "requires_actions": [],
        "status": "pending"
    },
    "resources": {
        "customer": {
            "id": "CU123",
            "created_at": "2021-03-22T12:20:04.238Z",
            "email": null,
            "given_name": "Alice",
            "family_name": "Smith",
            "company_name": null,
            "language": "en",
            "phone_number": null,
            "metadata": {}
        },
        "customer_billing_detail": {
            "id": "CBD123",
            "created_at": "2021-03-22T12:20:04.238Z",
            "address_line1": null,
            "address_line2": null,
            "address_line3": null,
            "city": null,
            "region": null,
            "postal_code": null,
            "country_code": null,
            "swedish_identity_number": null,
            "danish_identity_number": null
        }
    },
    "fallback_enabled": true,
    "fallback_occurred": true
}
}

```

PHP

Python

Ruby

Java

JavaScript

.NET

Go

## Change currency

This will allow for the updating of the currency and subsequently the scheme if needed for a Billing Request. This will only be available for mandate only flows which do not have the lock\_currency flag set to true on the Billing Request Flow. It will also not support any request which has a payments request.

Relative endpoint: POST /billing\_requests/BRQ123/actions/choose\_currency

Parameters

**currency**  
`required ISO 4217` currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.  
**metadata**  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

PHP

Python

Ruby

Java

JavaScript

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

.NET

Go

## Select institution for a Billing Request

Creates an Institution object and attaches it to the Billing Request

Relative endpoint: POST /billing\_requests/BRQ123/actions/select\_institution

**Restricted:** this endpoint is restricted to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with the [custom payment pages upgrade](#).

Parameters

country\_code  
*required* [ISO 3166-1](#) alpha-2 code. The country code of the institution. If nothing is provided, institutions with the country code 'GB' are returned by default.  
institution  
*required* The unique identifier for this institution

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/billing_requests/BRQ123/actions/select_institution HTTP/1.1
Content-Type: application/json
{
  "data": {
    "institution": "monzo",
    "country_code": "GB"
  }
}
```

HTTP/1.1 200
Content-Type: application/json

```
{
  "billing_request": {
    "id": "BRQ123",
    "created_at": "2021-03-22T12:20:04.397Z",
    "status": "pending",
    "mandate_request": {
      "currency": "GBP",
      "scheme": "bacs",
      "links": {}
    },
    "payment_request": {
      "description": "First Payment",
      "currency": "GBP",
      "amount": 500,
      "scheme": "faster_payments",
      "links": {}
    },
    "metadata": null,
    "links": {
      "customer": "CU123",
      "customer_billing_detail": "CBD123"
    },
    "creditor_name": "Mr Creditor",
    "actions": [
      {
        "type": "choose_currency",
        "required": true,
        "completes_actions": [],
        "requires_actions": [],
        "status": "completed"
      },
      {
        "type": "collect_customer_details",
        "required": true,
        "completes_actions": [],
        "requires_actions": [
          "choose_currency"
        ],
        "status": "pending",
        "collect_customer_details": {
          "incomplete_fields": {
            "customer": [
              "email",
              "given_name",
              "family_name"
            ],
            "customer_billing_detail": [
              "account_number",
              "card_type",
              "provider"
            ]
          }
        }
      },
      {
        "type": "select_institution",
        "required": false,
        "completes_actions": [
          "choose_currency",
          "collect_customer_details"
        ],
        "requires_actions": [],
        "status": "completed"
      }
    ]
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "type": "collect_bank_account",
    "required": true,
    "completes_actions": [
      "choose_currency"
    ],
    "requires_actions": [],
    "status": "pending"
  },
  {
    "type": "bank_authorisation",
    "required": true,
    "completes_actions": [],
    "requires_actions": [
      "collect_bank_account"
    ],
    "status": "pending"
  }
],
"resources": {
  "customer": {
    "id": "CU123",
    "created_at": "2021-03-22T12:20:04.238Z",
    "email": null,
    "given_name": "Alice",
    "family_name": "Smith",
    "company_name": null,
    "language": "en",
    "phone_number": null,
    "metadata": {}
  },
  "customer_billing_detail": {
    "id": "CBD123",
    "created_at": "2021-03-22T12:20:04.374Z",
    "address_line1": "1 Somewhere Lane",
    "address_line2": null,
    "address_line3": null,
    "city": null,
    "region": null,
    "postal_code": null,
    "country_code": null,
    "swedish_identity_number": null,
    "danish_identity_number": null
  },
  "institution": {
    "id": "monzo",
    "name": "Monzo",
    "icon_url": "https://images.example.com/image/332bb781-3cc2-4f3e-ae79-1aba09fac991?size",
    "roles": ["institutions_list", "heartbeat", "read_refund_account", "read_debtor_account"],
    "country_code": "GB"
  }
}
}
}
PHP

```

Python

Ruby

Java

JavaScript

.NET

Go

## Billing Request Flows

Billing Request Flows can be created to enable a payer to authorise a payment created for a scheme with strong payer authorisation (such as open banking single payments).

### Properties

<b>id</b>	Unique identifier, beginning with <code>BRF</code> .
<b>authorisation_url</b>	URL for a GC-controlled flow.
<b>auto_fulfil</b>	(Experimental feature) Fulfilment of the flow.
<b>created_at</b>	Timestamp when the flow was created.
<b>customer_details_captured</b>	Identifies whether a Billing Request Flow has captured customer details.
<b>exit_uri</b>	URL that the payer can be taken to if there isn't a way to progress ahead in flow.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Accepted currently.



Consultez notre politique de confidentialité

**expires\_at**

Timestamp when the flow will expire. Each flow currently lasts for 7 days.

**language**

Sets the default language of the Billing Request Flow and the customer. [ISO 639-1](#) code.

**lock\_bank\_account**

If true, the payer will not be able to change their bank account within the flow. If the bank\_account details are collected as part of bank\_authorisation then GC will set this value to true mid flow.

You can only lock bank account if these have already been completed as a part of the billing request.

**lock\_currency**

If true, the payer will not be able to change their currency/scheme manually within the flow. Note that this only applies to the mandate only flows - currency/scheme can never be changed when there is a specified subscription or payment.

**lock\_customer\_details**

If true, the payer will not be able to edit their customer details within the flow. If the customer details are collected as part of bank\_authorisation then GC will set this value to true mid flow.

You can only lock customer details if these have already been completed as a part of the billing request.

**prefilled\_bank\_account[account\_type]**

Bank account type for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.

**prefilled\_customer[address\_line1]**

The first line of the customer's address.

**prefilled\_customer[address\_line2]**

The second line of the customer's address.

**prefilled\_customer[address\_line3]**

The third line of the customer's address.

**prefilled\_customer[city]**

The city of the customer's address.

**prefilled\_customer[company\_name]**

Customer's company name. Company name should only be provided if given\_name and family\_name are null.

**prefilled\_customer[country\_code]**

[ISO 3166-1 alpha-2 code.](#)

**prefilled\_customer[danish\_identity\_number]**

For Danish customers only. The civic/company number (CPR or CVR) of the customer.

**prefilled\_customer[email]**

Customer's email address.

**prefilled\_customer[family\_name]**

Customer's surname.

**prefilled\_customer[given\_name]**

Customer's first name.

**prefilled\_customer[postal\_code]**

The customer's postal code.

**prefilled\_customer[region]**

The customer's address region, county or department.

**prefilled\_customer[swedish\_identity\_number]**

For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer.

**redirect\_uri**

URL that the payer can be redirected to after completing the request flow.

**session\_token**

Session token populated when responding to the initialise action

**show\_redirect\_buttons**

If true, the payer will be able to see redirect action buttons on Thank You page. These action buttons will provide a way to connect back to the billing request flow app if opened within a mobile app. For successful flow, the button will take the payer back the billing request flow where they will see the success screen. For failure, button will take the payer to url being provided against exit\_uri field.

**show\_success\_redirect\_button**

If true, the payer will be able to see a redirect action button on the Success page. This action button will provide a way to redirect the payer to the given redirect\_uri. This functionality is helpful when merchants do not want payers to be automatically redirected or on Android devices, where automatic redirections are not possible.

**skip\_success\_screen**

If true, the payer will not be redirected to the success screen after completing the flow. A redirect\_uri needs to be provided for this parameter to be taken into account.

**links[billing\_request]**

ID of the [billing request](#) against which this flow was created.

## Create a Billing Request Flow

Creates a new billing request flow.

Relative endpoint: POST /billing\_request\_flows

### Parameters

**auto\_fulfil**

(Experimental feature) Fulfill

**customer\_details\_captured**

Identifies whether a Billing

**exit\_uri**

URL that the payer can be t

**language**

Sets the default language of

**lock\_bank\_account**

If true, the payer will not be  
will set this value to true mi

You can only lock bank acc

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

llowed currently.

t of bank\_authorisation then GC



Consultez notre politique de confidentialité

**lock\_currency**

If true, the payer will not be able to change their currency/scheme manually within the flow. Note that this only applies to the mandate only flows - currency/scheme can never be changed when there is a specified subscription or payment.

**lock\_customer\_details**

If true, the payer will not be able to edit their customer details within the flow. If the customer details are collected as part of bank\_authorisation then GC will set this value to true mid flow.

You can only lock customer details if these have already been completed as a part of the billing request.

**prefilled\_bank\_account[account\_type]**

Bank account type for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local\\_details](#) for more information.

**prefilled\_customer[address\_line1]**

The first line of the customer's address.

**prefilled\_customer[address\_line2]**

The second line of the customer's address.

**prefilled\_customer[address\_line3]**

The third line of the customer's address.

**prefilled\_customer[city]**

The city of the customer's address.

**prefilled\_customer[company\_name]**

Customer's company name. Company name should only be provided if given\_name and family\_name are null.

**prefilled\_customer[country\_code]**

[ISO 3166-1 alpha-2 code](#).

**prefilled\_customer[danish\_identity\_number]**

For Danish customers only. The civic/company number (CPR or CVR) of the customer.

**prefilled\_customer[email]**

Customer's email address.

**prefilled\_customer[family\_name]**

Customer's surname.

**prefilled\_customer[given\_name]**

Customer's first name.

**prefilled\_customer[postal\_code]**

The customer's postal code.

**prefilled\_customer[region]**

The customer's address region, county or department.

**prefilled\_customer[swedish\_identity\_number]**

For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer.

**redirect\_uri**

URL that the payer can be redirected to after completing the request flow.

**show\_redirect\_buttons**

If true, the payer will be able to see redirect action buttons on Thank You page. These action buttons will provide a way to connect back to the billing request flow app if opened within a mobile app. For successful flow, the button will take the payer back the billing request flow where they will see the success screen. For failure, button will take the payer to url being provided against exit\_uri field.

**show\_success\_redirect\_button**

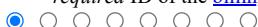
If true, the payer will be able to see a redirect action button on the Success page. This action button will provide a way to redirect the payer to the given redirect\_uri. This functionality is helpful when merchants do not want payers to be automatically redirected or on Android devices, where automatic redirections are not possible.

**skip\_success\_screen**

If true, the payer will not be redirected to the success screen after completing the flow. A redirect\_uri needs to be provided for this parameter to be taken into account.

**links[billing\_request]**

required ID of the [billing request](#) against which this flow was created.



HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/billing_request_flows HTTP/1.1
Content-Type: application/json
```

```
{
  "billing_request_flows": {
    "redirect_uri": "https://my-company.com/landing",
    "exit_uri": "https://my-company.com/exit",
    "prefilled_customer": {
      "given_name": "Frank",
      "family_name": "Osborne",
      "email": "frank.osborne@acmeplc.com"
    },
    "links": {
      "billing_request": "BRQ123"
    }
  }
}
```

HTTP/1.1 201 Created

Location: /billing\_request\_flows/BRF123456

Content-Type: application/json

```
{
  "billing_request_flows": {
    "id": "BRF123456",
    "auto_fulfil": true,
    "redirect_uri": "https://my-co",
    "exit_uri": "https://my-co",
    "authorisation_url": "http",
    "lock_customer_details": f,
    "lock_currency": false,
    "lock_bank_account": false,
    "skip_success_screen": fal,
    "session_token": null,
    "expires_at": "2021-07-30T",
    "created_at": "2021-07-23T",
    "links": {
      "billing_request": "BRQ123"
    },
    "show_redirect_buttons": false,
  }
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "show_success_redirect_button": false,
        "customer_details_captured": false,
    }
}
PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequestFlows()->create([
    "params" => [
        "redirect_uri" => "https://my-company.com/landing",
        "exit_uri"     => "https://my-company.com/exit",
        "prefilled_customer" => [
            "given_name" => "Frank",
            "family_name" => "Osborne",
            "email" => "frank.osborne@acmeplc.com"
        ],
        "links" => [
            "billing_request" => "BRQ123"
        ]
    ]
]);

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_request_flows.create(params={
    "redirect_uri": "https://my-company.com/landing",
    "exit_uri": "https://my-company.com/exit",
    "prefilled_customer": {
        "given_name": "Frank",
        "family_name": "Osborne",
        "email": "frank.osborne@acmeplc.com"
    },
    "links": {
        "billing_request": "BRQ123"
    }
})

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_request_flows.create(
  params: {
    redirect_uri: "https://my-company.com/landing",
    exit_uri: "https://my-company.com/exit",
    prefilled_customer: {
      given_name: "Frank",
      family_name: "Osborne",
      email: "frank.osborne@acmeplc.com"
    },
    links: {
      billing_request: "BRQ123",
    }
  }
)

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

BillingRequestFlow billingRequestFlow = client.billingRequestFlows().create()
    .withRedirectUri("https://my-company.com/landing")
    .withExitUri("https://my-company.com/exit")
    .withPrefilledCustomerGivenName("Frank")
    .withPrefilledCustomerFamilyName("Osborne")
    .withPrefilledCustomerEmail("frank.osborne@acmeplc.com")
    .withLinksBillingRequest("BRQ123")
    .execute();

```

## JavaScript

```

const constants = require('gocardless-nodeis/constants');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const billingRequestFlow = await client.billingRequestFlows().create({
    redirect_uri: "https://my-company.com/landing",
    exit_uri: "https://my-company.com/exit",
    prefilled_customer: {
        given_name: "Frank",
        family_name: "Osborne",
        email: "frank.osborne@acmeplc.com"
    },
    links: {
        billing_request: "BRQ123",
    }
});

```

## .NET

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var prefilledCustomer = new GoCardless.Services.RedirectFlowCreateRequest.RedirectFlowPrefilledCustomer()
{
    AddressLine1 = "338 Goswell Road",
    Email = "api@gocardless.com",
    GivenName = "Bobby",
    FamilyName = "Tables"
};

var resp = await gocardless.BillingRequestFlows.CreateAsync(
    new GoCardless.Services.BillingRequestFlowCreateRequest()
    {
        RedirectUri = "https://my-company.com/landing",
        ExitUri = "https://my-company.com/exit",
        PrefilledCustomer = prefilledCustomer,
        Links = new gocardless.BillingRequestFlowCreateRequest.BillingRequestFlowLinks
        {
            BillingRequest = "BRQ123",
        },
    }
);

GoCardless.Resources.BillingRequestFlow billingRequestFlow = resp.BillingRequestFlow;
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestFlowCreateParams := gocardless.BillingRequestFlowCreateParams{
        RedirectUri: "https://my-company.com/landing",
        ExitUri: "https://my-company.com/exit",
        PrefilledCustomer: &gocardless.RedirectFlowCreateParamsPrefilledCustomer{
            AddressLine1: "338-346 Goswell Road",
            City: "London",
            GivenName: "Tim",
            FamilyName: "Rogers",
            Email: "tim@gocardless.com",
            PostalCode: "EC1V 7LQ",
        },
        Links: gocardless.BillingRequestFlowCreateParamsLinks{
            BillingRequest: "BR123",
        },
    }
    billingRequestFlow, err := client.BillingRequestFlows.Create(context, billingRequestFlowCreateParams)
}
}

```

## Initialise a Billing Request Flow

Returns the flow having generated a fresh session token which can be used to power integrations that manipulate the flow.

Relative endpoint: POST /billing\_request\_flows/BRF123/actions initialise

Parameters



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com/billing\_request\_flows/BRF123/actions/initialise HTTP/1.1

HTTP/1.1 200  
Content-Type: application/json

```
{
    "billing_request_flows": {
        "id": "BRF123",
        "auto_fulfil": true,
        "redirect_uri": null,
        "exit_uri": null,
        "authorisation_url": "http://",
        "lock_customer_details": false,
        "lock_bank_account": false,
        "lock_currency": false,
        "skip_success_screen": false,
        "session_token": "seshsai",
        "expires_at": "2021-09-01T12:00:00Z",
        "created_at": "2021-08-25T12:00:00Z",
        "links": {
            "billing_request": "BRQ123",
        },
        "show_redirect_buttons": true,
        "show_success_redirect_buttons": false,
        "customer_details_captured": false
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    }
}
PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
$client->billingRequestFlows()->initialise("BRF123");

Python

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
client.billing_request_flows.initialise("BRF123")

Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_request_flows.initialise("BRF123")

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.billingRequestFlows().initialise("BRF123").execute();

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.billingRequestFlows.initialise("BRF123");

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequestFlows.InitialiseAsync("BRF123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestFlowInitialiseParams := gocardless.BillingRequestFlowInitialiseParams{}
    billingRequestFlow, err := client.BillingRequestFlows.Initialise(context, "BRF123", billingRequestFlowInitialiseParams)
}

}

```

## Billing Request Templates

Billing Request Templates are reusable templates that result in numerous Billing Requests with similar attributes. They provide a no-code solution for generating various types of multi-user payment links.

Each template includes a reusable URL that can be embedded in a website or shared with customers via email. Every time the URL is opened, it generates a new Billing Request.

Billing Request Templates overcome the single-use nature of standard payment links and is designed to cater for multiple users.

### Properties

<b>id</b>	Unique identifier, beginning with <code>BR</code> .
<b>authorisation_url</b>	Permanent URL that customers can use to make payments.
<b>created_at</b>	Fixed <code>timestamp</code> , recording when the request was created.
<b>mandate_request_constraints[end_date]</b>	The latest date at which payments can be taken, must occur after <code>start_date</code> if present.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

sers because it is intended for

redirect\_uri.

This is an optional field and if it is not supplied the agreement will be considered open and will not have an end date. Keep in mind the end date must take into account how long it will take the user to set up this agreement via the Billing Request.

#### `mandate_request_constraints[max_amount_per_payment]`

The maximum amount that can be charged for a single payment. Required for PayTo and VRP.

#### `mandate_request_constraints[payment_method]`

A constraint where you can specify info (free text string) about how payments are calculated. *Note:* This is only supported for ACH and PAD schemes.

#### `periodic_limits`

List of periodic limits and constraints which apply to them

Each instance will contain these properties:

- `alignment`: The alignment of the period.

`calendar` - this will finish on the end of the current period. For example this will expire on the Monday for the current week or the January for the next year.

`creation_date` - this will finish on the next instance of the current period. For example Monthly it will expire on the same day of the next month, or yearly the same day of the next year.

- `max_payments`: (Optional) The maximum number of payments that can be collected in this periodic limit.

- `max_total_amount`: The maximum total amount that can be charged for all payments in this periodic limit. Required for VRP.

- `period`: The repeating period for this mandate. Defaults to flexible for PayTo if not specified.

#### `mandate_request_constraints[start_date]`

The date from which payments can be taken.

This is an optional field and if it is not supplied the start date will be set to the day authorisation happens.

#### `mandate_request_currency`

[ISO 4217 currency code](#).

#### `mandate_request_description`

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

#### `mandate_request_metadata`

Key-value store of custom data that will be applied to the mandate created when this request is fulfilled. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### `mandate_request_scheme`

A bank payment scheme. Currently “ach”, “autogiro”, “baes”, “becs”, “becs\_nz”, “betalingsservice”, “faster\_payments”, “pad”, “pay\_to” and “sepa\_core” are supported. Optional for mandate only requests - if left blank, the payer will be able to select the currency/scheme to pay with from a list of your available schemes.

#### `mandate_request_verify`

Verification preference for the mandate. One of:

- `minimum`: only verify if absolutely required, such as when part of scheme rules
- `recommended`: in addition to `minimum`, use the GoCardless payment intelligence solution to decide if a payer should be verified
- `when_available`: if verification mechanisms are available, use them
- `always`: as `when_available`, but fail to create the Billing Request if a mechanism isn't available

By default, all Billing Requests use the recommended verification preference. It uses GoCardless payment intelligence solution to determine if a payer is fraudulent or not. The verification mechanism is based on the response and the payer may be asked to verify themselves. If the feature is not available, `recommended` behaves like `minimum`.

If you never wish to take advantage of our reduced risk products and Verified Mandates as they are released in new schemes, please use the `minimum` verification preference.

See [Billing Requests: Creating Verified Mandates](#) for more information.

#### `metadata`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### `name`

Name for the template. Provides a friendly human name for the template, as it is shown in the dashboard. Must not exceed 255 characters.

#### `payment_request_amount`

Amount in full.

#### `payment_request_currency`

[ISO 4217 currency code](#). GBP and EUR supported; GBP with your customers in the UK and for EUR with your customers in supported Eurozone countries only.

#### `payment_request_description`

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

#### `payment_request_metadata`

Key-value store of custom data that will be applied to the payment created when this request is fulfilled. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### `payment_request_scheme`

(Optional) A scheme used for sepa\_instant\_credit\_trans.

Please be aware that sepa\_i

#### `redirect_uri`

URL that the payer can be re

#### `updated_at`

Dynamic [timestamp](#) recordin

### List Billing Request Templa

Returns a [cursor-paginated](#) list of

Relative endpoint: GET /billing\_r

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

`edit_transfer` and `transfer` is used as the default.



Consultez notre politique de confidentialité

## Parameters

**after**  
Cursor pointing to the start of the desired set.

**before**  
Cursor pointing to the end of the desired set.

**limit**  
Number of records to return.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/billing_request_templates HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "billing_request_templates": [
    {
      "id": "BRT123",
      "name": "12 Month Gold Plan",
      "mandate_request_currency": "GBP",
      "mandate_request_scheme": null,
      "mandate_request_verify": null,
      "mandate_request_metadata": null,
      "payment_request_description": "One-time joining fee",
      "mandate_request_description": "Recurring fee",
      "mandate_request_constraints": {
        "max_amount_per_payment": 1000
      },
      "payment_request_amount": "69.99",
      "payment_request_currency": "GBP",
      "payment_request_scheme": null,
      "payment_request_metadata": null,
      "redirect_uri": "https://my-company.com/landing",
      "authorisation_url": "https://pay.gocardless.com/BRT123",
      "metadata": {
        "checkout_flow": "primary"
      },
      "created_at": "2021-07-27T17:10:33.784Z",
      "updated_at": "2021-07-27T17:10:33.784Z"
    }
  ]
}
```

## PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->billingRequestTemplates()->list();
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_request_templates.list().records
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.billing_request_templates.list
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
for (BillingRequestTemplate bi : client.billing_request_templates.list()) {
  System.out.println(billingRe...}
```

## JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');
const billingRequestTemplates = client.billing_request_templates;
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var billingRequestTemplateListResponse = gocardless.BillingRequestTemplates.All();
foreach (GoCardless.Resources.BillingRequestTemplate billingRequestTemplate in billingRequestTemplateListResponse)
{
    Console.WriteLine(billingRequestTemplate.Id);
}
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestTemplateListParams := gocardless.BillingRequestTemplateListParams{}
    billingRequestTemplateListResult, err := client.BillingRequestTemplates.List(context, billingRequestTemplateListParams)
    for _, billingRequestTemplate := range billingRequestTemplateListResult.BillingRequestTemplates {
        fmt.Println(billingRequestTemplate.Id)
    }
}

```

## Get a single Billing Request Template

Fetches a Billing Request Template

Relative endpoint: GET /billing\_request\_templates/BRT123



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/billing\_request\_templates/BRT123 HTTP/1.1

HTTP/1.1 200  
Content-Type: application/json

```
{
    "billing_request_templates": {
        "id": "BRT123",
        "name": "12 Month Gold Plan",
        "mandate_request_currency": "GBP",
        "mandate_request_scheme": null,
        "mandate_request_verify": null,
        "mandate_request_metadata": null,
        "payment_request_description": "One-time joining fee",
        "mandate_request_description": "Recurring fee",
        "mandate_request_constraints": {
            "max_amount_per_payment": 1000
        },
        "payment_request_amount": "69.99",
        "payment_request_currency": "GBP",
        "payment_request_scheme": null,
        "payment_request_metadata": null,
        "redirect_uri": "https://my-company.com/landing",
        "authorisation_url": "https://pay.gocardless.com/BRT123",
        "metadata": {
            "checkout-flow": "primary"
        },
        "created_at": "2021-07-27T17:10:33.784Z",
        "updated_at": "2021-07-27T17:10:33.784Z"
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->billingRequestTemplates()->get("BRT123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client
```

```
client.billing_request_template
```

Ruby

```
@client = GoCardlessPro::Client.new(
    access_token: "your_access_token_here",
    environment: :sandbox
)
```

```
@client.billing_request_template
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.billingRequestTemplates().get("BRT123").execute();
JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const resp = await client.billingRequestTemplates.find("BRT123");

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequestTemplates.GetAsync("BRT123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestTemplate, err := client.BillingRequestTemplates.Get(context, "BRT123")
}

```

## Create a Billing Request Template

Relative endpoint: POST /billing\_request\_templates

Parameters

`mandate_request_constraints[end_date]`

The latest date at which payments can be taken, must occur after start\_date if present

This is an optional field and if it is not supplied the agreement will be considered open and will not have an end date. Keep in mind the end date must take into account how long it will take the user to set up this agreement via the Billing Request.

`mandate_request_constraints[max_amount_per_payment]`

The maximum amount that can be charged for a single payment. Required for PayTo and VRP.

`mandate_request_constraints[payment_method]`

A constraint where you can specify info (free text string) about how payments are calculated. *Note:* This is only supported for ACH and PAD schemes.

`mandate_request_constraints[periodic_limits]`

List of periodic limits and constraints which apply to them

`mandate_request_constraints[start_date]`

The date from which payments can be taken.

This is an optional field and if it is not supplied the start date will be set to the day authorisation happens.

`mandate_request_currency`

[ISO 4217](#) currency code.

`mandate_request_description`

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

`mandate_request_metadata`

Key-value store of custom data that will be applied to the mandate created when this request is fulfilled. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`mandate_request_scheme`

A bank payment scheme. Currently “ach”, “autogiro”, “baes”, “becs”, “becs\_nz”, “betalingsservice”, “faster\_payments”, “pad”, “pay\_to” and “sepa\_core” are supported. Optional for mandates. From the payment scheme dropdown, select the payment scheme to use, from a list of your available schemes.

`mandate_request_verify`

Verification preference for the mandate.

- `minimum`: only verify if the mandate is used for the first time.
- `recommended`: in addition to minimum, verify if the mandate is used for recurring payments.
- `when_available`: if verification is available, verify the mandate.
- `always`: as when\_available, but always verify the mandate.

By default, all Billing Requests use the `minimum` verification mechanism. The verification mechanism is determined by the value of `mandate.verify`.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

ifified

determine if a payer is fraudulent or liable, recommended behaves like

If you never wish to take advantage of our reduced risk products and Verified Mandates as they are released in new schemes, please use the `minimum` verification preference.

See [Billing Requests: Creating Verified Mandates](#) for more information.

#### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### name

Name for the template. Provides a friendly human name for the template, as it is shown in the dashboard. Must not exceed 255 characters.

#### payment\_request\_amount

Amount in full.

#### payment\_request\_currency

[ISO 4217](#) currency code. GBP and EUR supported; GBP with your customers in the UK and for EUR with your customers in supported Eurozone countries only.

#### payment\_request\_description

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

#### payment\_request\_metadata

Key-value store of custom data that will be applied to the payment created when this request is fulfilled. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### payment\_request\_scheme

(Optional) A scheme used for Open Banking payments. Currently `faster_payments` is supported in the UK (GBP) and `sepa_credit_transfer` and `sepa_instant_credit_transfer` are supported in supported Eurozone countries (EUR). For Eurozone countries, `sepa_credit_transfer` is used as the default. Please be aware that `sepa_instant_credit_transfer` may incur an additional fee for your customer.

#### redirect\_uri

URL that the payer can be redirected to after completing the request flow.

#### links[creditor]

ID of the associated [creditor](#). Only required if your account manages multiple creditors.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/billing_request_templates HTTP/1.1
Content-Type: application/json
```

```
{
```

```
  "billing_request_templates": {
    "name": "12 Month Gold Plan",
    "payment_request_description": "One-time joining fee",
    "mandate_request_description": "Recurring fee",
    "mandate_request_constraints": {
      "max_amount_per_payment": 1000
    },
    "payment_request_currency": "GBP",
    "payment_request_amount": "69.99",
    "mandate_request_currency": "GBP",
    "redirect_uri": "https://my-company.com/landing",
    "metadata": {
      "checkout-flow": "primary"
    }
  }
}
```

HTTP/1.1 201 Created

Location: /billing\_request\_templates/BRT123

Content-Type: application/json

```
{
```

```
  "billing_request_templates": {
    "id": "BRT123",
    "name": "12 Month Gold Plan",
    "mandate_request_currency": "GBP",
    "mandate_request_scheme": null,
    "mandate_request_verify": null,
    "mandate_request_metadata": null,
    "payment_request_description": "One-time joining fee",
    "mandate_request_description": "Recurring fee",
    "mandate_request_constraints": {
      "max_amount_per_payment": 1000
    },
    "payment_request_amount": "69.99",
    "payment_request_currency": "GBP",
    "payment_request_scheme": null,
    "payment_request_metadata": null,
    "redirect_uri": "https://my-company.com/landing",
    "authorisation_url": "https://pay.gocardless.com/BRT123",
    "metadata": {
      "checkout-flow": "primary"
    },
    "created_at": "2021-07-27T17:10:33.784Z",
    "updated_at": "2021-07-27T17:10:33.784Z"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token => 'your_access_token';
$environment  => \GoCardlessPro\Environment::Production();
]);
$client->billingRequestTemplate([
  'params' => [
    'name' => '12 Month Gold Plan',
    'payment_request_description' => 'One-time joining fee',
    'mandate_request_description' => 'Recurring fee',
    'payment_request_currency' => 'GBP',
    'payment_request_amount' => '69.99',
    'mandate_request_currency' => 'GBP',
    'redirect_uri' => "https://my-company.com/landing"
  ]
]);
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_request_templates.create(params={
    "name": "12 Month Gold Plan",
    "payment_request_description": "One-time joining fee",
    "mandate_request_description": "Recurring fee",
    "payment_request_currency": "GBP",
    "payment_request_amount": "69.99",
    "mandate_request_currency": "GBP",
    "redirect_uri": "https://my-company.com/landing",
})
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_request_templates.create(
  params: {
    name: "12 Month Gold Plan",
    payment_request_description: "One-time joining fee",
    mandate_request_description: "Recurring fee",
    payment_request_currency: "GBP",
    payment_request_amount: "69.99",
    mandate_request_currency: "GBP",
    redirect_uri: "https://my-company.com/landing",
  }
)
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

BillingRequestTemplate billingRequestTemplate = client.billingRequestTemplates().create()
  .withPaymentRequestDescription("One-time joining fee")
  .withPaymentRequestCurrency("GBP")
  .withPaymentRequestAmount("69.99")
  .withMandateRequestCurrency("GBP")
  .withRedirectUri("https://my-company.com/landing")
  .execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const billingRequestTemplate = await client.billingRequestTemplates.create({
  name: "12 Month Gold Plan",
  payment_request_description: "One-time joining fee",
  mandate_request_description: "Recurring fee",
  payment_request_currency: "GBP",
  payment_request_amount: "69.99",
  mandate_request_currency: "GBP",
  redirect_uri: "https://my-company.com/landing",
});
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequestTemplates.CreateAsync(
  new GoCardless.Services.BillingRequestTemplateCreateRequest()
  {
    PaymentRequestDescription = "One-time joining fee",
    PaymentRequestCurrency = "GBP",
    PaymentRequestAmount = 6999,
    MandateRequestCurrency = "GBP",
    RedirectUri = "https://my-company.com/landing",
  }
);
```

```
GoCardless.Resources.BillingRequestTemplate billingRequestTemplate = resp.BillingRequestTemplate;
```

## Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig(accessToken)
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in init: %v", err)
    return
  }
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

billingRequestTemplateCreateParams := gocardless.BillingRequestTemplateCreateParams{
    Name: "12 Month Gold Plan",
    PaymentRequestDescription: "One-time joining fee",
    PaymentRequestCurrency: "GBP",
    PaymentRequestAmount: 6999,
    MandateRequestCurrency: "GBP",
    RedirectUri: "https://my-company.com/landing",
}

billingRequestTemplate, err := client.BillingRequestTemplates.Create(context, billingRequestTemplateCreateParams)
}

```

## Update a Billing Request Template

Updates a Billing Request Template, which will affect all future Billing Requests created by this template.

Relative endpoint: PUT /billing\_request\_templates/BRQ123

### Parameters

`mandate_request_constraints[end_date]`

The latest date at which payments can be taken, must occur after `start_date` if present

This is an optional field and if it is not supplied the agreement will be considered open and will not have an end date. Keep in mind the end date must take into account how long it will take the user to set up this agreement via the Billing Request.

`mandate_request_constraints[max_amount_per_payment]`

The maximum amount that can be charged for a single payment. Required for PayTo and VRP.

`mandate_request_constraints[payment_method]`

A constraint where you can specify info (free text string) about how payments are calculated. *Note:* This is only supported for ACH and PAD schemes.

`mandate_request_constraints[periodic_limits]`

List of periodic limits and constraints which apply to them

`mandate_request_constraints[start_date]`

The date from which payments can be taken.

This is an optional field and if it is not supplied the start date will be set to the day authorisation happens.

`mandate_request_currency`

[ISO 4217](#) currency code.

`mandate_request_description`

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

`mandate_request_metadata`

Key-value store of custom data that will be applied to the mandate created when this request is fulfilled. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`mandate_request_scheme`

A bank payment scheme. Currently “ach”, “autogiro”, “bacs”, “becs”, “becs\_nz”, “betalingsservice”, “faster\_payments”, “pad”, “pay\_to” and “sepa\_core” are supported. Optional for mandate only requests - if left blank, the payer will be able to select the currency/scheme to pay with from a list of your available schemes.

`mandate_request_verify`

Verification preference for the mandate. One of:

- `minimum`: only verify if absolutely required, such as when part of scheme rules
- `recommended`: in addition to `minimum`, use the GoCardless payment intelligence solution to decide if a payer should be verified
- `when_available`: if verification mechanisms are available, use them
- `always`: as `when_available`, but fail to create the Billing Request if a mechanism isn't available

By default, all Billing Requests use the recommended verification preference. It uses GoCardless payment intelligence solution to determine if a payer is fraudulent or not. The verification mechanism is based on the response and the payer may be asked to verify themselves. If the feature is not available, `recommended` behaves like `minimum`.

If you never wish to take advantage of our reduced risk products and Verified Mandates as they are released in new schemes, please use the `minimum` verification preference.

See [Billing Requests: Creating Verified Mandates](#) for more information.

`metadata`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`name`

Name for the template. Provides a friendly human name for the template, as it is shown in the dashboard. Must not exceed 255 characters.

`payment_request_amount`

Amount in full.

`payment_request_currency`

[ISO 4217](#) currency code. GE

`payment_request_description`

A human-readable description of the payment and/or mandate.

`payment_request_metadata`

Key-value store of custom data that will be applied to the mandate created when this request is fulfilled. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`payment_request_scheme`

(Optional) A scheme used for the mandate. Currently “ach”, “autogiro”, “bacs”, “becs”, “becs\_nz”, “betalingsservice”, “faster\_payments”, “pad”, “pay\_to” and “sepa\_core” are supported. Optional for mandate only requests - if left blank, the payer will be able to select the currency/scheme to pay with from a list of your available schemes.

Please be aware that `sepa_instant_credit_transfer` is only supported in the Eurozone countries.

`redirect_uri`

URL that the payer can be redirected to after they have completed the payment process.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

rted Eurozone countries only.

; request.

nitted, with key names up to 50

edit\_transfer and transfer is used as the default.

[HTTP](#) [PHP](#) [Python](#) [Ruby](#) [Java](#) [JavaScript](#) [.NET](#) [Go](#)
[HTTP](#)

```
PUT https://api.gocardless.com/billing_request_templates/BRT123 HTTP/1.1
Content-Type: application/json
{
  "billing_request_templates": {
    "name": "12 Month Silver Plan",
    "payment_request_amount": "49.99"
  }
}

HTTP/1.1 200 OK
Location: /billing_request_templates/BRT123
Content-Type: application/json
{
  "billing_request_templates": {
    "id": "BRT123",
    "name": "12 Month Silver Plan",
    "mandate_request_currency": "GBP",
    "mandate_request_scheme": null,
    "mandate_request_verify": null,
    "mandate_request_metadata": null,
    "payment_request_description": "One-time joining fee",
    "mandate_request_description": "Recurring fee",
    "mandate_request_constraints": {
      "max_amount_per_payment": 1000
    },
    "payment_request_amount": "49.99",
    "payment_request_currency": "GBP",
    "payment_request_scheme": null,
    "payment_request_metadata": null,
    "redirect_uri": "https://my-company.com/landing",
    "authorisation_url": "https://pay.gocardless.com/BRT123",
    "metadata": {
      "checkout-flow": "primary"
    },
    "created_at": "2021-07-27T17:10:33.784Z",
    "updated_at": "2021-07-30T11:28:11.123Z"
  }
}
```

**PHP**

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->billingRequestTemplates()->update("BRT123", [
  "params" => [
    "name" => "12 Month Silver Plan",
    "payment_request_amount" => "49.99"
  ]
]);
```

**Python**

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.billing_request_templates.update("BRT123", params={
  "name": "12 Month Silver Plan",
  "payment_request_amount": "49.99",
})
```

**Ruby**

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.billing_request_templates.update("BRT123", params: {
  name: "12 Month Silver Plan",
  payment_request_amount: "49.99",
})
```

**Java**

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
BillingRequestTemplate billing =
  .withName("12 Month Silver P")
  .withPaymentRequestAmount("49.99")
  .execute();
```

**JavaScript**

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const billingRequestTemplate =
  { name: "12 Month Silver Plan",
    payment_request_amount: "49.99" };
});
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.BillingRequestTemplates.UpdateAsync(
    "BRT123",
    new GoCardless.Services.BillingRequestTemplateUpdateRequest()
{
    Name = "12 Month Silver Plan",
    PaymentRequestAmount = 4999,
}
);

GoCardless.Resources.BillingRequestTemplate billingRequestTemplate = resp.BillingRequestTemplate;

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    billingRequestTemplateUpdateParams := gocardless.BillingRequestTemplateUpdateParams{
        Name:           "12 Month Silver Plan",
        PaymentRequestAmount: 4999,
    }

    billingRequestTemplate, err := client.BillingRequestTemplates.Update(context, "BRT123", billingRequestTemplateUpdateParams)
}
}

```

## Billing Request with Actions

Billing Requests help create resources that require input or action from a customer. An example of required input might be additional customer billing details, while an action would be asking a customer to authorise a payment using their mobile banking app.

See [Billing Requests: Overview](#) for how-to's, explanations and tutorials.

### Properties

`bank_authorisations[authorisation_type]`  
 Type of authorisation, can be either 'mandate' or 'payment'.

`bank_authorisations[authorised_at]`  
 Fixed [timestamp](#), recording when the user has been authorised.

`bank_authorisations[created_at]`  
 Timestamp when the flow was created

`bank_authorisations[expires_at]`  
 Timestamp when the url will expire. Each authorisation url currently lasts for 15 minutes, but this can vary by bank.

`bank_authorisations[id]`  
 Unique identifier, beginning with "BAU".

`bank_authorisations[last_visited_at]`  
 Fixed [timestamp](#), recording when the authorisation URL has been visited.

`links[billing_request]`  
 ID of the [billing request](#) against which this authorisation was created.

`links[institution]`  
 ID of the [institution](#) against which this authorisation was created.

`bank_authorisations[qr_code_url]`  
 URL to a QR code PNG image of the bank authorisation url. This QR code can be used as an alternative to providing the `url` to the payer to allow them to authorise with their mobile devices.

`bank_authorisations[redirect_uri]`  
 URL that the payer can be redirected to after authorising the payment.

On completion of bank authorisation, the query parameter of either `outcome=success` or `outcome=failure` will be appended to the `redirect_uri` to indicate the result of the bank authorisation. If the bank authorisation is expired, the query parameter `outcome=timeout` will be appended to the `redirect_uri`, in which case you should prompt the user to

Please note: bank authorisations are triggered by a bank authorisation event, such as a mandate or payment.

The BillingRequestFlow ID

Defaults to <https://pay.gocardless.com>

`bank_authorisations[url]`  
 URL for an oauth flow that

`actions`  
 List of actions that can be performed

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

wait for the relevant bank authorisation to show the correct outcome

Each instance will contain these properties:

- `available_currencies`: List of currencies the current mandate supports
- `bank_authorisation`: Describes the behaviour of bank authorisations, for the `bank_authorisation` action
- `collect_customer_details`: Additional parameters to help complete the `collect_customer_details` action
- `completes_actions`: Which other action types this action can complete.
- `institution_guess_status`: Describes whether we inferred the institution from the provided bank account details. One of:
  - `not_needed`: we won't attempt to infer the institution as it is not needed. Either because it was manually selected or the billing request does not support this feature
  - `pending`: we are waiting on the bank details in order to infer the institution
  - `failed`: we weren't able to infer the institution
  - `success`: we inferred the institution and added it to the resources of a Billing Request
- `required`: Informs you whether the action is required to fulfil the billing request or not.
- `requires_actions`: Requires completing these actions before this action can be completed.
- `status`: Status of the action
- `type`: Unique identifier for the action.

`billing_requests[created_at]`Fixed [timestamp](#), recording when this resource was created.`billing_requests[fallback_enabled]`

(Optional) If true, this billing request can fallback from instant payment to direct debit. Should not be set if GoCardless payment intelligence feature is used.

See [Billing Requests: Retain customers with Fallbacks](#) for more information.`billing_requests[fallback_occurred]`

True if the billing request was completed with direct debit.

`billing_requests[id]`

Unique identifier, beginning with "BRQ".

`instalment_schedule_request[app_fee]`

The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

`instalment_schedule_request[currency]`[ISO 4217](#) currency code. Currently "USD" and "CAD" are supported.`instalments_with_dates`An explicit array of instalment payments, each specifying at least an `amount` and `charge_date`. See [create \(with dates\)](#)

Each instance will contain these properties:

- `amount`: Amount, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
- `charge_date`: A future date on which the payment should be collected. If the date is before the `next_possible_charge_date` on the [mandate](#), it will be automatically rolled forwards to that date.
- `description`: A human-readable description of the payment. This will be included in the notification email GoCardless sends to your customer if your organisation does not send its own notifications (see [compliance requirements](#)).

`amounts`

List of amounts of each instalment, in the lowest denomination for the currency (e.g. cents in USD).

`instalments_with_schedule[interval]`Number of `interval_units` between charge dates. Must be greater than or equal to 1.`instalments_with_schedule[interval_unit]`The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.`instalments_with_schedule[start_date]`The date on which the first payment should be charged. Must be on or after the [mandate](#)'s `next_possible_charge_date`. When left blank and `month` or `day_of_month` are provided, this will be set to the date of the first payment. If created without `month` or `day_of_month` this will be set as the mandate's `next_possible_charge_date``links[instalment_schedule]`(Optional) ID of the [instalment\\_schedule](#) that was created from this instalment schedule request.`instalment_schedule_request[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`instalment_schedule_request[name]`

Name of the instalment schedule, up to 100 chars. This name will also be copied to the payments of the instalment schedule if you use schedule-based creation.

`instalment_schedule_request[payment_reference]`An optional payment reference. This will be set as the reference on each payment created and will appear on your customer's bank statement. See the documentation for the [create payment endpoint](#) for more details.`instalment_schedule_request[retry_if_possible]`On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).`instalment_schedule_request[total_amount]`

The total amount of the instalments (in cents in EUR). If the request

currency (e.g. pence in GBP,

`links[bank_authorisation]`(Optional) ID of the [bank account](#)`links[creditor]`ID of the associated [credito](#)`links[customer]`ID of the [customer](#) that will receive the payment`links[customer_bank_account]`(Optional) ID of the [customer bank account](#)`links[customer_billing_detail]`

ID of the customer billing detail

`links[instalment_schedule_request]`(Optional) ID of the associated [instalment schedule request](#)

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

`links[instalment_schedule_request_instalment_schedule]`  
 (Optional) ID of the [instalment\\_schedule](#) that was created from this instalment schedule request.

`links[mandate_request]`  
 (Optional) ID of the associated mandate request

`links[mandate_request_mandate]`  
 (Optional) ID of the [mandate](#) that was created from this mandate request. this mandate request.

`links[organisation]`  
 ID of the associated organisation.

`links[payment_provider]`  
 (Optional) ID of the associated payment provider

`links[payment_request]`  
 (Optional) ID of the associated payment request

`links[payment_request_payment]`  
 (Optional) ID of the [payment](#) that was created from this payment request.

`links[subscription_request]`  
 (Optional) ID of the associated subscription request

`links[subscription_request_subscription]`  
 (Optional) ID of the [subscription](#) that was created from this subscription request.

`mandate_request[authorisation_source]`  
 This field is ACH specific, sometimes referred to as [SEC code](#).

This is the way that the payer gives authorisation to the merchant. web: Authorisation is Internet Initiated or via Mobile Entry (maps to SEC code: WEB) telephone: Authorisation is provided orally over telephone (maps to SEC code: TEL) paper: Authorisation is provided in writing and signed, or similarly authenticated (maps to SEC code: PPD)

`mandate_request[consent_type]`  
 This attribute represents the authorisation type between the payer and merchant. It can be set to `one_off`, `recurring` or `standing` for ACH scheme. And `single`, `recurring` and `sporadic` for PAD scheme. *Note:* This is only supported for ACH and PAD schemes.

`constraints[end_date]`  
 The latest date at which payments can be taken, must occur after `start_date` if present

This is an optional field and if it is not supplied the agreement will be considered open and will not have an end date. Keep in mind the end date must take into account how long it will take the user to set up this agreement via the Billing Request.

`constraints[max_amount_per_payment]`  
 The maximum amount that can be charged for a single payment. Required for PayTo and VRP.

`constraints[payment_method]`  
 A constraint where you can specify info (free text string) about how payments are calculated. *Note:* This is only supported for ACH and PAD schemes.

`periodic_limits`  
 List of periodic limits and constraints which apply to them

Each instance will contain these properties:

- `alignment`: The alignment of the period.  
`calendar` - this will finish on the end of the current period. For example this will expire on the Monday for the current week or the January for the next year.
- `creation_date` - this will finish on the next instance of the current period. For example Monthly it will expire on the same day of the next month, or yearly the same day of the next year.
- `max_payments`: (Optional) The maximum number of payments that can be collected in this periodic limit.
- `max_total_amount`: The maximum total amount that can be charged for all payments in this periodic limit. Required for VRP.
- `period`: The repeating period for this mandate. Defaults to flexible for PayTo if not specified.

`constraints[start_date]`

The date from which payments can be taken.

This is an optional field and if it is not supplied the start date will be set to the day authorisation happens.

`mandate_request[currency]`

[ISO 4217](#) currency code.

`mandate_request[description]`

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

`links[mandate]`

(Optional) ID of the [mandate](#) that was created from this mandate request. this mandate request.

`mandate_request[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`mandate_request[payer_requested_dual_signature]`

This attribute can be set to true if the payer has indicated that multiple signatures are required for the mandate. As long as every other Billing Request actions have been completed, the payer will receive an email notification containing instructions on how to complete the additional signature. The dual signature flow can only be completed using GoCardless branded pages.

`mandate_request[scheme]`

A bank payment scheme. Currently supported. Optional for most schemes.

`mandate_request[sweeping]`

If true, this billing request will sweep funds from the bank account that the payer also owns. This is currently not supported.

`mandate_request[verify]`

Verification preference for this mandate.

- `minimum`: only verify if there is a payment due.
- `recommended`: in addition to minimum, verify if the mandate is used.
- `when_available`: if verification mechanisms are available, use them

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

”, “pay\_to” and “sepa\_core” are from a list of your available

1 by the payer to another account

ified

- always: as when\_available, but fail to create the Billing Request if a mechanism isn't available

By default, all Billing Requests use the recommended verification preference. It uses GoCardless payment intelligence solution to determine if a payer is fraudulent or not. The verification mechanism is based on the response and the payer may be asked to verify themselves. If the feature is not available, recommended behaves like minimum.

If you never wish to take advantage of our reduced risk products and Verified Mandates as they are released in new schemes, please use the minimum verification preference.

See [Billing Requests: Creating Verified Mandates](#) for more information.

#### billing\_requests[metadata]

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### payment\_request[amount]

Amount in minor unit (e.g. pence in GBP, cents in EUR).

#### payment\_request[app\_fee]

The amount to be deducted from the payment as an app fee, to be paid to the partner integration which created the billing request, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

#### payment\_request[currency]

[ISO 4217](#) currency code. GBP and EUR supported; GBP with your customers in the UK and for EUR with your customers in supported Eurozone countries only.

#### payment\_request[description]

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

#### payment\_request[funds\_settlement]

This field will decide how GoCardless handles settlement of funds from the customer.

- managed will be moved through GoCardless' account, batched, and payed out.
- direct will be a direct transfer from the payer's account to the merchant where invoicing will be handled separately.

#### links[payment]

(Optional) ID of the [payment](#) that was created from this payment request.

#### payment\_request[metadata]

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### payment\_request[reference]

A custom payment reference defined by the merchant. It is only available for payments on the PayTo scheme or payments using the Direct Funds settlement model on the Faster Payments scheme.

#### payment\_request[scheme]

(Optional) A scheme used for Open Banking payments. Currently faster\_payments is supported in the UK (GBP) and sepa\_credit\_transfer and sepa\_instant\_credit\_transfer are supported in supported Eurozone countries (EUR). For Eurozone countries, sepa\_credit\_transfer is used as the default. Please be aware that sepa\_instant\_credit\_transfer may incur an additional fee for your customer.

#### billing\_requests[purpose\_code]

Specifies the high-level purpose of a mandate and/or payment using a set of pre-defined categories. Required for the PayTo scheme, optional for all others. Currently mortgage, utility, loan, dependant\_support, gambling, retail, salary, personal, government, pension, tax and other are supported.

#### customer[company\_name]

Customer's company name. Required unless a given\_name and family\_name are provided. For Canadian customers, the use of a company\_name value will mean that any mandate created from this customer will be considered to be a "Business PAD" (otherwise, any mandate will be considered to be a "Personal PAD").

#### customer[created\_at]

Fixed [timestamp](#), recording when this resource was created.

#### customer[email]

Customer's email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

#### customer[family\_name]

Customer's surname. Required unless a company\_name is provided.

#### customer[given\_name]

Customer's first name. Required unless a company\_name is provided.

#### customer[id]

Unique identifier, beginning with "CU".

#### customer[language]

[ISO 639-1](#) code. Used as the language for notification emails sent by GoCardless if your organisation does not send its own (see [compliance requirements](#)).

Currently only "en", "fr", "de", "pt", "es", "it", "nl", "da", "nb", "sl", "sv" are supported. If this is not provided, the language will be chosen based on the country\_code (if supplied) or default to "en".

#### customer[metadata]

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### customer[phone\_number]

[ITU E.123](#) formatted phone number, including country code.

#### customer\_bank\_account[account\_holder\_name]

Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a [customer bank account token](#).

#### customer\_bank\_account[account\_numberEnding]

The last few digits of the account number. Currently 4 digits for NZD bank accounts and 2 digits for other currencies.

#### customer\_bank\_account[account\_type]

Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.

#### customer\_bank\_account[bank\_account\_token]

A token to uniquely refer to a set of bank account details. This feature is still in early access and is only available for certain organisations.

#### customer\_bank\_account[bank\_name]

Name of bank, taken from the bank details

#### customer\_bank\_account[country]

[ISO 3166-1 alpha-2 code](#)

#### customer\_bank\_account[created\_at]

Fixed [timestamp](#), recording

#### customer\_bank\_account[currency]

[ISO 4217](#) currency code. C

#### customer\_bank\_account[enabled]

Boolean value showing wh

#### customer\_bank\_account[id]

Unique identifier, beginning

#### links[customer]

ID of the [customer](#) that own

#### customer\_bank\_account[metadata]

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

`customer_billing_detail[address_line1]`  
The first line of the customer's address.

`customer_billing_detail[address_line2]`  
The second line of the customer's address.

`customer_billing_detail[address_line3]`  
The third line of the customer's address.

`customer_billing_detail[city]`  
The city of the customer's address.

`customer_billing_detail[country_code]`  
[ISO 3166-1 alpha-2 code](#).

`customer_billing_detail[created_at]`  
Fixed [timestamp](#), recording when this resource was created.

`customer_billing_detail[danish_identity_number]`  
For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer's bank account is denominated in Danish krone (DKK).

`customer_billing_detail[id]`  
Unique identifier, beginning with "CU".

`customer_billing_detail[ip_address]`  
For ACH customers only. Required for ACH customers. A string containing the IP address of the payer to whom the mandate belongs (i.e. as a result of their completion of a mandate setup flow in their browser).

Not required for creating offline mandates where `authorisation_source` is set to telephone or paper.

`customer_billing_detail[postal_code]`  
The customer's postal code.

`customer_billing_detail[region]`  
The customer's address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

`schemes`  
The schemes associated with this customer billing detail

`customer_billing_detail[swedish_identity_number]`  
For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer's bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.

`billing_requests[status]`  
One of:

- `pending`: the billing request is pending and can be used
- `ready_to_fulfil`: the billing request is ready to fulfil
- `fulfilling`: the billing request is currently undergoing fulfilment
- `fulfilled`: the billing request has been fulfilled and a payment created
- `cancelled`: the billing request has been cancelled and cannot be used

`subscription_request[amount]`  
Amount in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

`subscription_request[app_fee]`  
The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

`subscription_request[count]`  
The total number of payments that should be taken by this subscription.

`subscription_request[currency]`  
[ISO 4217](#) currency code. Currently "USD" and "CAD" are supported.

`subscription_request[day_of_month]`  
As per RFC 2445. The day of the month to charge customers on. 1-28 or -1 to indicate the last day of the month.

`subscription_request[interval]`  
Number of `interval_units` between customer charge dates. Must be greater than or equal to 1. Must result in at least one charge date per year. Defaults to 1.

`subscription_request[interval_unit]`  
The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.

`links[subscription]`  
(Optional) ID of the [subscription](#) that was created from this subscription request.

`subscription_request[metadata]`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`subscription_request[month]`  
Name of the month on which to charge a customer. Must be lowercase. Only applies when the `interval_unit` is `yearly`.

`subscription_request[name]`  
Optional name for the subscription. This will be set as the description on each payment created. Must not exceed 255 characters.

`subscription_request[payment_reference]`  
An optional payment reference. This will be set as the reference on each payment created and will appear on your customer's bank statement. See the documentation for the [create payment endpoint](#) for more details.

`subscription_request[retry_if_possible]`  
On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dash](#)'.

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Relative endpoint: POST /billing.

**Note:** to modify data on an already

**Restricted:** this endpoint is restricted to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with the [custom payment pages upgrade](#).

**Warning:** ACH/PAD is not supported on this endpoint.

#### Parameters

`actions[bank_authorisation_redirect_uri]`

URL for an oauth flow that will allow the user to authorise the payment

`actions[collect_bank_account]`

#### Properties:

- `account_holder_name`: Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a [customer bank account token](#).
- `account_number`: Bank account number - see [local details](#) for more information. Alternatively you can provide an [iban](#).
- `account_number_suffix`: Account number suffix (only for bank accounts denominated in NZD) - see [local details](#) for more information.
- `account_type`: Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.
- `bank_code`: Bank code - see [local details](#) for more information. Alternatively you can provide an [iban](#).
- `branch_code`: Branch code - see [local details](#) for more information. Alternatively you can provide an [iban](#).
- `country_code`: [ISO 3166-1 alpha-2 code](#). Defaults to the country code of the [iban](#) if supplied, otherwise is required.
- `currency`: [ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.
- `iban`: International Bank Account Number. Alternatively you can provide [local details](#). IBANs are not accepted for Swedish bank accounts denominated in SEK - you must supply [local details](#).
- `metadata`: Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.
- `pay_id`: A unique record such as an email address, mobile number or company number, that can be used to make and accept payments.

`actions[collect_customer_details]`

#### Properties:

- `customer`:
- `customer_billing_detail`:

`actions[confirm_payer_details]`

#### Properties:

- `metadata`: Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.
- `payer_requested_dual_signature`: This attribute can be set to true if the payer has indicated that multiple signatures are required for the mandate. As long as every other Billing Request actions have been completed, the payer will receive an email notification containing instructions on how to complete the additional signature. The dual signature flow can only be completed using GoCardless branded pages.

`actions[create_bank_authorisation]`

Create a bank authorisation object as part of this request

`actions[select_institution]`

#### Properties:

- `country_code`: [ISO 3166-1](#) alpha-2 code. The country code of the institution. If nothing is provided, institutions with the country code ‘GB’ are returned by default.
- `institution`: The unique identifier for this institution

`fallback_enabled`

(Optional) If true, this billing request can fallback from instant payment to direct debit. Should not be set if GoCardless payment intelligence feature is used.

See [Billing Requests: Retain customers with Fallbacks](#) for more information.

`mandate_request[authorisation_source]`

This field is ACH specific, sometimes referred to as [SEC code](#).

This is the way that the payer gives authorisation to the merchant. web: Authorisation is Internet Initiated or via Mobile Entry (maps to SEC code: WEB) telephone: Authorisation is provided orally over telephone (maps to SEC code: TEL) paper: Authorisation is provided in writing and signed, or similarly authenticated (maps to SEC code: PPD)

`mandate_request[constraints]`

Constraints that will apply to the mandate\_request. (Optional) Specifically required for PayTo and VRP.

#### Properties:

- `end_date`: The latest date at which payments can be taken, must occur after start\_date if present

This is an optional field and if it is not supplied the agreement will be considered open and will not have an end date. Keep in mind the end date must take into account how long it will take the user to set up this agreement via the Billing Request.

- `max_amount_per_payment`
- `payment_method`: A code for PAD schemes.
- `periodic_limits`: List of periodic limits.
- `start_date`: The date when the mandate begins.

This is an optional field.

`mandate_request[currency]`

[ISO 4217](#) currency code.

`mandate_request[description]`

A human-readable description.

`mandate_request[metadata]`

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



is only supported for ACH and



Consultez notre politique de confidentialité

; request.

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### `mandate_request[reference]`

Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

#### `mandate_request[scheme]`

A bank payment scheme. Currently “ach”, “autogiro”, “baes”, “becs”, “becs\_nz”, “betalingsservice”, “faster\_payments”, “pad”, “pay\_to” and “sepa\_core” are supported. Optional for mandate only requests - if left blank, the payer will be able to select the currency/scheme to pay with from a list of your available schemes.

#### `mandate_request[sweeping]`

If true, this billing request would be used to set up a mandate solely for moving (or sweeping) money from one account owned by the payer to another account that the payer also owns. This is required for Faster Payments

#### `mandate_request[verify]`

Verification preference for the mandate. One of:

- `minimum`: only verify if absolutely required, such as when part of scheme rules
- `recommended`: in addition to `minimum`, use the GoCardless payment intelligence solution to decide if a payer should be verified
- `when_available`: if verification mechanisms are available, use them
- `always`: as `when_available`, but fail to create the Billing Request if a mechanism isn't available

By default, all Billing Requests use the recommended verification preference. It uses GoCardless payment intelligence solution to determine if a payer is fraudulent or not. The verification mechanism is based on the response and the payer may be asked to verify themselves. If the feature is not available, `recommended` behaves like `minimum`.

If you never wish to take advantage of our reduced risk products and Verified Mandates as they are released in new schemes, please use the `minimum` verification preference.

See [Billing Requests: Creating Verified Mandates](#) for more information.

#### `metadata`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### `payment_request[amount]`

Amount in minor unit (e.g. pence in GBP, cents in EUR).

#### `payment_request[app_fee]`

The amount to be deducted from the payment as an app fee, to be paid to the partner integration which created the billing request, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

#### `payment_request[currency]`

[ISO 4217](#) currency code. `GBP` and `EUR` supported; `GBP` with your customers in the UK and for `EUR` with your customers in supported Eurozone countries only.

#### `payment_request[description]`

A human-readable description of the payment and/or mandate. This will be displayed to the payer when authorising the billing request.

#### `payment_request[funds_settlement]`

This field will decide how GoCardless handles settlement of funds from the customer.

- `managed` will be moved through GoCardless' account, batched, and payed out.
- `direct` will be a direct transfer from the payer's account to the merchant where invoicing will be handled separately.

#### `payment_request[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### `payment_request[reference]`

A custom payment reference defined by the merchant. It is only available for payments on the PayTo scheme or payments using the Direct Funds settlement model on the Faster Payments scheme.

#### `payment_request[retry_if_possible]`

On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#). **Important:** This is not applicable to IBP and VRP payments.

#### `payment_request[scheme]`

(Optional) A scheme used for Open Banking payments. Currently `faster_payments` is supported in the UK (GBP) and `sepa_credit_transfer` and `sepa_instant_credit_transfer` are supported in supported Eurozone countries (EUR). For Eurozone countries, `sepa_credit_transfer` is used as the default. Please be aware that `sepa_instant_credit_transfer` may incur an additional fee for your customer.

#### `purpose_code`

Specifies the high-level purpose of a mandate and/or payment using a set of pre-defined categories. Required for the PayTo scheme, optional for all others. Currently `mortgage`, `utility`, `loan`, `dependant_support`, `gambling`, `retail`, `salary`, `personal`, `government`, `pension`, `tax` and `other` are supported.

#### `links[creditor]`

ID of the associated [creditor](#). Only required if your account manages multiple creditors.

#### `links[customer]`

ID of the [customer](#) against which this request should be made.

#### `links[customer_bank_account]`

(Optional) ID of the [customer\\_bank\\_account](#) against which this request should be made.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/billing_requests/create_with_actions HTTP/1.1
Content-Type: application/json
{
```

```
  "billing_request_with_actions": {
    "payment_request": {
      "description": "First Pay",
      "amount": "500",
      "currency": "GBP"
    },
    "actions": {
      "collect_customer_detail": {
        "customer": {
          "email": "alice@example.com",
          "given_name": "Alice",
          "family_name": "Smith"
        }
      },
      "select_institution": {
        "institution": "IN123",
        "country_code": "GB"
      }
    }
  }
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
        },
        "create_bank_authorisation": true
    }
}

HTTP/1.1 200 OK
Content-Type: application/json
{
    "billing_request_with_actions": {
        "billing_requests": {
            "id": "BRQ123",
            "created_at": "2025-03-06T21:36:43Z",
            "status": "pending",
            "payment_request": {
                "description": "First Payment",
                "amount": "500",
                "currency": "GBP"
            },
            "actions": [
                {
                    "type": "bank_authorisation",
                    "required": true,
                    "status": "pending"
                }
            ],
            "resources": {
                "customer": {
                    "id": "CU123",
                    "email": "alice@example.com",
                    "given_name": "Alice",
                    "family_name": "Smith",
                    "company_name": null,
                    "language": "en",
                    "phone_number": null,
                    "created_at": "2025-03-06T21:36:43Z"
                }
            },
            "links": {
                "customer": "CU123"
            }
        },
        "bank_authorisations": {
            "id": "BAU123",
            "created_at": "2025-03-06T21:36:43Z",
            "expires_at": "2025-03-06T21:51:43Z",
            "url": "https://pay.gocardless.com/obauth/BAU123",
            "qr_code_url": "https://pay.gocardless.com/obauth/BAU123/qr_code",
            "authorisation_type": "payment",
            "links": {
                "billing_request": "BRQ123"
            }
        }
    }
}

```

PHP

Python

Ruby

Java

JavaScript

.NET

Go

## Institutions

Institutions that are supported when using the GoCardless API.

Not all institutions support both payment and collection.

Properties

**id**

The unique identifier for this institution. This is used to collect bank account details from GoCardless.

**autocomplete\_collect\_bank\_account**  
Flag to show if selecting this institution allows GoCardless to collect bank account details to GoCardless.

**country\_code**  
[ISO 3166-1](#) alpha-2 code. This is used to collect bank account details from GoCardless.

**icon\_url**  
A URL pointing to the icon for this institution.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

the bank can return the payer's

re returned by default.

Consultez notre politique de confidentialité

```

limits[daily]
  Daily limit details for this institution. (The 'limits' property is only available via an authenticated request with a generated access token)
limits[single]
  Single transaction limit details for this institution. (The 'limits' property is only available via an authenticated request with a generated access token)
logo_url
  A URL pointing to the logo for this institution
name
  A human readable name for this institution
status
  The status of the institution

```

## List Institutions

Returns a list of supported institutions.

Relative endpoint: GET /institutions

Parameters

```

branch_code
  (Currently only supports UK sort-codes) The six-digit number that identifies both the bank and the specific branch where an account is held, eg. '601234'.
country_code
  ISO 3166-1 alpha-2 code. The country code of the institution. If nothing is provided, institutions with the country code 'GB' are returned by default.
feature
  The feature that institutions support. The available options include pis, and vrp_sweeping. If nothing is provided, institutions supporting 'pis' are returned by default.
scheme
  The scheme that institutions support. The available options include faster_payments, sepa_credit_transfer, and sepa_instant_credit_transfer. If nothing is provided, institutions supporting 'faster_payments' are returned by default.

```



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/institutions?country\_code=GB HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

{

```

  "institutions": [
    {
      "id": "monzo",
      "name": "Monzo",
      "logo_url": "https://assets.gocardless.com/icon",
      "icon_url": "https://assets.gocardless.com/icon",
      "country_code": "GB",
      "limits": {
        "daily": {
          "business": "5000000",
          "personal": "2000000"
        },
        "single": {
          "business": "5000000",
          "personal": "2000000"
        }
      }
    }
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->institutions()->list([
  'params' => ["country_code" => "GB"]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

@client.institutions.list(params={"country_code": "GB"}).records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

@client.institutions.list(params:

Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient.newBuilder()
  .accessToken(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

Institution institution = client.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

// List of all institutions.
const institutions = await client.institutions.list();

// List of institutions for a country code.
const institutions = await client.institutions.list({ countryCode: 'GB' });

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

// List of institutions for a country code.
var request = new GoCardless.Services.InstitutionListRequest()
{
    CountryCode = "GB",
};

var institutionListResponse = gocardless.Institutions.All(request);

// List of all institutions.
var institutionListResponse = gocardless.Institutions.All();

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    institutionListParams := gocardless.InstitutionListParams{}
    institutionListResult, err := client.Institutions().List(context, institutionListParams)
    for _, institution := range institutionListResult.Institutions {
        fmt.Println(institution.Id)
    }
}

}
```

**List institutions for Billing Request**

Returns all institutions valid for a Billing Request.

This endpoint is currently supported only for FasterPayments.

Relative endpoint: GET /billing\_requests/BRQ123/institutions

**Restricted:** this endpoint is restricted to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with the [custom payment pages upgrade](#).

## Parameters

`country_code`  
`required ISO 3166-1` alpha-2 code. The country code of the institution. If nothing is provided, institutions with the country code 'GB' are returned by default.

`ids`  
ID(s) of the institution(s) to retrieve. More than one ID can be specified using a comma-separated string.

`include_disabled`  
Indicates whether to include temporarily disabled institutions in the response. If not provided or set to false, only enabled institutions will be returned.

`search`  
A search substring for retrieving institution(s), based on the institution's name.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/

HTTP/1.1 200  
Content-Type: application/json

```
{
    "institutions": [
        {
            "id": "BANK_OF_SCOTLAND_____",
            "name": "Bank of Scotland",
            "logo_url": "https://c...",
            "icon_url": "https://c...",
            "country_code": "GB",
            "autocomplete_collect": true,
            "status": "enabled"
        },
        {
            "id": "BANK_OF_SCOTLAND_____",
            "name": "Bank of Scotland Personal",
        }
    ]
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "logo_url": "https://cdn-logos.gocardless.com/ais/BANK_OF_SCOTLAND_BUSINESS_BOFGBS1.png",
        "icon_url": "https://cdn-logos.gocardless.com/ais/BANK_OF_SCOTLAND_BUSINESS_BOFGBS1.png",
        "country_code": "GB",
        "autocomplete_collect_bank_account": true,
        "status": "temporarily_disabled"
    }
]
}
PHP

```

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->institutions()->listForBillingRequest("BR123", [
    "params" => ["country_code" => "GB"]
]);

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

@client.institutions.listForBillingRequest("BR123", params={
    "country_code": "GB"
}).records

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.institutions.listForBillingRequest("BR123", {
  params: { country_code: "GB" }
})

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.institutions().listForBillingRequest("BR123").withCountryCode("GB").execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const institutions = await client.institutions.list_for_billing_request("BR123", { countryCode: 'GB'});

```

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var request = new GoCardless.Services.InstitutionListForBillingRequestRequest()
{
    CountryCode = "GB",
};

var institutionListResponse = await gocardless.Institutions.ListForBillingRequestAsync("BR123", request);

```

## Go

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in init")
        return
    }

    institutionListForBillingReq
    CountryCode: "GB",
}
institutionListForBillingReq
for _, institution := range :
    fmt.Println(institution.
}
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

tionListForBillingRequestParams)

## Core Endpoints



Consultez notre politique de confidentialité

## Balances

Returns the balances for a creditor. These balances are the same as what's shown in the dashboard with one exception (mentioned below under `balance_type`).

These balances will typically be 3-5 minutes old. The balance amounts likely won't match what's shown in the dashboard as the dashboard balances are updated much less frequently (once per day).

### Properties

#### amount

The total amount in the balance, defined as the sum of all debits subtracted from the sum of all credits, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

#### balance\_type

Type of the balance. Could be one of

- `pending_payments_submitted`: Payments we have submitted to the scheme but not yet confirmed. This does not exactly correspond to *Pending payments* in the dashboard, because this balance does not include payments that are pending submission.
- `confirmed_funds`: Payments that have been confirmed minus fees and unclaimed debits for refunds, failures and chargebacks. These funds have not yet been moved into a payout.
- `pending_payouts`: Confirmed payments that have been moved into a payout. This is the total due to be paid into your bank account in the next payout run (payouts happen once every business day). `pending_payouts` will only be non-zero while we are generating and submitting the payouts to our partner bank.

#### currency

[ISO 4217](#) currency code. Currently "AUD", "CAD", "DKK", "EUR", "GBP", "NZD", "SEK" and "USD" are supported.

#### last\_updated\_at

Dynamic [timestamp](#) recording when this resource was last updated.

#### links[creditor]

ID of the associated [creditor](#).

## List balances

Returns a [cursor-paginated](#) list of balances for a given creditor. This endpoint is rate limited to 60 requests per minute.

Relative endpoint: GET /balances

### Parameters

#### creditor

required ID of a [creditor](#).

#### after

Cursor pointing to the start of the desired set.

#### before

Cursor pointing to the end of the desired set.

#### limit

Number of records to return.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/balances?creditor=CR123 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "balances": [
    {
      "balance_type": "pending_payments_submitted",
      "amount": 365000,
      "currency": "EUR",
      "last_updated_at": "2025-03-17T11:08:04.926Z",
      "links": {
        "creditor": "CR123"
      }
    },
    {
      "balance_type": "pending_payouts",
      "amount": 0,
      "currency": "EUR",
      "last_updated_at": "2025-03-17T11:48:14.416Z",
      "links": {
        "creditor": "CR123"
      }
    },
    {
      "balance_type": "confirmed_funds",
      "amount": 421198,
      "currency": "EUR",
      "last_updated_at": "2025-03-17T11:48:14.416Z",
      "links": {
        "creditor": "CR123"
      }
    }
  ],
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    }
  },
  "limit": 50
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->balances()->list([
    'params' => [
        'creditor' => "CR123",
    ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.balances.list(params={
    "creditor": "CR123",
}).records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.balances.list(
  params: {
    creditor: "CR123",
  }
).records
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.balances().
    all().
    withCreditor("CR123").
    execute()
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

await client.balances.list({
    creditor: "CR123",
});
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var request = new GoCardless.Services.BalancesListRequest()
{
    Creditor = "CR123",
};

gocardless.Balances.All(request);
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.NewClient(config)
    if err != nil {
        fmt.Println("error in init")
        return
    }

    params := gocardless.BalanceParams{
        Creditor: "CR123",
    }

    response, err := client.Balances.All(params)
    if err != nil {
        fmt.Println("error in listing")
        return
    }

    fmt.Println(response)
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Bank Account Details

Retrieve bank account details in JWE encrypted format

#### Properties

```
ciphertext
  Base64 URL encoded encrypted payload, in this case bank details.
encrypted_key
  Base64 URL encoded symmetric content encryption key, encrypted with the asymmetric key from your JWKS.
iv
  Base64 URL encoded initialization vector, used during content encryption.
protected
  Base64 URL encoded JWE header values, containing the following keys:
    • alg: the asymmetric encryption type used to encrypt symmetric key, e.g: RSA-OAEP.
    • enc: the content encryption type, e.g: A256GCM.
    • kid: the ID of an RSA-2048 public key, from your JWKS, used to encrypt the AES key.
tag
  Base64 URL encoded authentication tag, used to verify payload integrity during decryption.
```

### Get encrypted bank details

Returns bank account details in the flattened JSON Web Encryption format described in RFC 7516

Relative endpoint: GET /bank\_account\_details/BA123

**Note:** See our [Security Requirements](#) before using this feature

**Restricted:** This endpoint is restricted to organisations with the Encrypted Bank Details Access upgrade

#### Parameters

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/bank_account_details/BA123 HTTP/1.1
Content-Type: application/json
```

HTTP/1.1 200 (OK)

Location: /bank\_account\_details/BA123

Content-Type: application/json

```
{
  "bank_account_details": {
    "protected": "eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkEyNTZhQ00iLCJraWQiOjIy2RkNDkzMnJmx0DY0MjI3YjhkZEZiMDdkMjQ5MDgyOGEzYwQ50WM10TyxNzc3ZDbjZTMzMDUy
    "encrypted_key": "pEQHnx_jCjKLiHzMa27C8cXgIlu7qUdrEcPuRwQw9EZyA_3vHG99gdd7YGQapVbSGEG43Hg5gi33Sgjsuekp6CumRFoBIKR7mANK5LUGfy6jFVFAC98RMiJ2H8yTBjLR
    "iv": "6H4MLnhzGnwaWLQ2",
    "ciphertext": "Cntt_wZlkK4Ui7xI-BtAqf9V-FF41Cz8RxJd5KUGQJac",
    "tag": "WoaSncwUeLfU3IjgeFYtSQ"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->bankAccountDetails()->get("BA123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.bank_account_details.get("BA123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.bank_account_details.get("BA123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
client.bankAccountDetails().get("BA123");
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');
const resp = await client.bank_account_details.get("BA123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);
```

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var resp = await gocardless.BankAccountDetails.GetAsync("BA123");
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    bankAccountDetails, err := client.BankAccountDetails.Get(context, "BA123")
}

```

## Blocks

Blocks are created to prevent certain customer details from being used when creating mandates.

The details used to create blocks can be exact matches, like a bank account or an email, or a more generic match such as an email domain or bank name. Please be careful when creating blocks for more generic matches as this may block legitimate payers from using your service.

New block types may be added over time.

A block is in essence a simple rule that is used to match against details in a newly created mandate. If there is a successful match then the mandate is transitioned to a “blocked” state.

Please note:

- Payments and subscriptions cannot be created against a mandate in blocked state.
- A mandate can never be transitioned out of the blocked state.

The one exception to this is when blocking a ‘bank\_name’. This block will prevent bank accounts from being created for banks that match the given name. To ensure we match bank names correctly an existing bank account must be used when creating this block. Please be aware that we cannot always match a bank account to a given bank name.

This API is currently only available for GoCardless Protect+ integrators - please [get in touch](#) if you would like to use this API.

Properties

**id**  
Unique identifier, beginning with “BLC”.

**active**  
Shows if the block is active or disabled. Only active blocks will be used when deciding if a mandate should be blocked.

**block\_type**  
Type of entity we will seek to match against when blocking the mandate. This can currently be one of ‘email’, ‘email\_domain’, ‘bank\_account’, or ‘bank\_name’.

**created\_at**  
Fixed [timestamp](#), recording when this resource was created.

**reason\_description**  
This field is required if the reason\_type is other. It should be a description of the reason for why you wish to block this payer and why it does not align with the given reason\_types. This is intended to help us improve our knowledge of types of fraud.

**reason\_type**  
The reason you wish to block this payer, can currently be one of ‘identity\_fraud’, ‘no\_intent\_to\_pay’, ‘unfair\_chargeback’. If the reason isn’t captured by one of the above then ‘other’ can be selected but you must provide a reason description.

**resource\_reference**  
This field is a reference to the value you wish to block. This may be the raw value (in the case of emails or email domains) or the ID of the resource (in the case of bank accounts and bank names). This means in order to block a specific bank account (even if you wish to block generically by name) it must already have been created as a resource.

**updated\_at**  
Fixed [timestamp](#), recording when this resource was updated.

## Create a block

Creates a new Block of a given type

Relative endpoint: POST /blocks

**Note:** Block will be enabled if you

Parameters

**active**

Shows if the block is active

**block\_type**

Type of entity we will seek

‘bank\_name’.

**reason\_description**

### ... Details >

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

’, ‘bank\_account’, or



Consultez notre politique de confidentialité

This field is required if the reason\_type is other. It should be a description of the reason for why you wish to block this payer and why it does not align with the given reason\_types. This is intended to help us improve our knowledge of types of fraud.

**reason\_type**

The reason you wish to block this payer, can currently be one of 'identity\_fraud', 'no\_intent\_to\_pay', 'unfair\_chargeback'. If the reason isn't captured by one of the above then 'other' can be selected but you must provide a reason description.

**resource\_reference**

This field is a reference to the value you wish to block. This may be the raw value (in the case of emails or email domains) or the ID of the resource (in the case of bank accounts and bank names). This means in order to block a specific bank account (even if you wish to block generically by name) it must already have been created as a resource.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/blocks HTTP/1.1
Content-Type: application/json
{
  "blocks": {
    "block_type": "email",
    "reason_type": "no_intent_to_pay",
    "resource_reference": "example@example.com"
  }
}
```

```
HTTP/1.1 201 Created
Location: /block/BLC123
Content-Type: application/json
{
  "blocks": {
    "id": "BLC123",
    "block_type": "email",
    "reason_type": "no_intent_to_pay",
    "resource_reference": "example@example.com",
    "active": "true",
    "created_at": "2021-03-25T17:26:28.305Z",
    "updated_at": "2021-03-25T17:26:28.305Z"
  }
}
```

```
POST https://api.gocardless.com/blocks HTTP/1.1
Content-Type: application/json
{
  "blocks": {
    "block_type": "bank_account",
    "reason_type": "no_intent_to_pay",
    "resource_reference": "BA123"
  }
}
```

```
HTTP/1.1 201 Created
Location: /block/BLC456
Content-Type: application/json
{
  "blocks": {
    "id": "BLC456",
    "block_type": "bank_account",
    "reason_type": "no_intent_to_pay",
    "resource_reference": "BA123",
    "active": "true",
    "created_at": "2021-03-25T17:26:28.305Z",
    "updated_at": "2021-03-25T17:26:28.305Z"
  }
}
```

**PHP**

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$block = $client->blocks()->create([
  'params' => [
    'block_type' => "email",
    'reason_type' => "no_intent_to_pay",
    'resource_reference' => "example@example.com",
  ]
]);
```

**Python**

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

block = client.blocks.create(
  params={
    "block_type": "email",
    "reason_type": "no_intent_to_pay",
    "resource_reference": "example@example.com"
  }
)
```

**Ruby**

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
block = @client.blocks.create(
  params: {
    block_type: "email",
    reason_type: "no_intent_to_pay",
  }
)
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        resource_reference: "example@example.com"
    }
}
Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

Block block = client.blocks().create()
    .withBlockType(Block.BlockType.EMAIL)
    .withReasonType(Block.ReasonType.NO_INTENT_TO_PAY)
    .withResourceReference("example@example.com")
    .execute();

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const block = await client.blocks.create({
    block_type: Types.BlockBlockType.Email,
    resource_reference: 'CU123',
    reason_type: Types.BlockReasonType.NoIntentToPay,
})

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.Blocks.CreateAsync(
    new GoCardless.Services.BlockCreateRequest()
{
    BlockType = GoCardless.Services.BlockCreateRequest.BlockBlockType.Email,
    ReasonType = GoCardless.Services.BlockCreateRequest.BlockReasonType.NoIntentToPay,
    ResourceReference = "example@example.com"
});
GoCardless.Resources.Block block = resp.Block;

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    blockCreateParams := gocardless.BlockCreateParams{
        BlockType:          "email",
        ReasonType:         "no_intent_to_pay",
        ResourceReference: "example@example.com",
    }

    block, err := client.Blocks.Create(context, blockCreateParams)
}

}

```

## Get a single block

Retrieves the details of an existing block.

Relative endpoint: GET /blocks/BLC123



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/
HTTP/1.1 200 OK
Content-Type: application/json
{
  "blocks": [
    {
      "id": "BLC456",
      "block_type": "email",
      "reason_type": "no_intent_to_pay",
      "resource_reference": "example@example.com",
      "active": "true",
      "created_at": "2021-03-25T14:52:00Z",
      "updated_at": "2021-03-25T14:52:00Z"
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$block = $client->blocks()->get("BLC456");
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

block = client.blocks.get("BLC456")
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

block = @client.blocks.get("BLC456")
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

Block block = client.blocks().get("BLC456").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const block = await client.blocks.find("BLC456");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.Blocks.GetAsync("BLC456");
GoCardless.Resources.Block block = resp.Block;
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    block, err := client.Blocks.Get(context, "BLC456")
}
```

**List multiple blocks**

Returns a [cursor-paginated](#) list of your blocks.

Relative endpoint: GET /blocks

Parameters

<code>after</code>	Cursor pointing to the start of the page.
<code>before</code>	Cursor pointing to the end of the page.
<code>block</code>	ID of a <a href="#">Block</a> .
<code>block_type</code>	Type of entity we will seek, e.g. 'bank_name'.
<code>created_at</code>	Fixed <a href="#">timestamp</a> , recording the creation date.
<code>limit</code>	Number of records to return.
<code>reason_type</code>	Reason type for the block.

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

', 'bank\_account', or



Consultez notre politique de confidentialité

The reason you wish to block this payer, can currently be one of 'identity\_fraud', 'no\_intent\_to\_pay', 'unfair\_chargeback'. If the reason isn't captured by one of the above then 'other' can be selected but you must provide a reason description.

**updated\_at**

Fixed [timestamp](#), recording when this resource was updated.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

**GET** <https://api.gocardless.com/blocks> HTTP/1.1

```
HTTP/1.1 200 OK
Location: /blocks
Content-Type: application/json
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "blocks": [
    {
      "id": "BLC123",
      "block_type": "email",
      "reason_type": "no_intent_to_pay",
      "resource_reference": "example@example.com",
      "active": "true",
      "created_at": "2021-03-25T17:26:28.305Z",
      "updated_at": "2021-03-25T17:26:28.305Z"
    },
    {
      "id": "BLC456",
      "block_type": "email_domain",
      "reason_type": "identity_fraud",
      "resource_reference": "example.com",
      "active": "true",
      "created_at": "2021-03-25T17:26:28.305Z",
      "updated_at": "2021-03-25T17:26:28.305Z"
    }
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

foreach ($client->blocks()->all() as $record) {
  echo $record->id;
  echo $record->block_type;
}
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

for block in client.blocks.all():
    print(block.id)
    print(block.block_type)
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.blocks.all.each do |block|
  puts block.id
  puts block.block_type
end
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

for (Block block : client.blocks().execute())
  System.out.println(block.get_id());
  System.out.println(block.get_type());
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

for await (const block of client.blocks().execute())
  console.log(block.id);
  console.log(block.block_type);
```

.NET

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

foreach (GoCardless.Resources.Block block in gocardless.Blocks.All())
{
    Console.WriteLine(block.Id);
}
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    blockListParams := gocardless.BlockListParams{}
    blockListResult, err := client.Blocks.List(context, blockListParams)
    for _, block := range blockListResult.Blocks {
        fmt.Println(block.Id)
    }
}

```

## Disable a block

Disables a block so that it no longer will prevent mandate creation.

Relative endpoint: POST /blocks/BLC123/actions/disable



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com/blocks/BLC123/actions/disable HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "blocks": [
        {
            "id": "BLC123",
            "block_type": "bank_account",
            "reason_type": "no_intent_to_pay",
            "resource_reference": "BA123",
            "active": "false",
            "created_at": "2021-03-25T17:26:28.305Z",
            "updated_at": "2021-03-25T19:26:28.305Z"
        }
    ]
}
```

POST https://api.gocardless.com/blocks/BLC123/actions/disable HTTP/1.1  
HTTP/1.1 409 Conflict

POST https://api.gocardless.com/blocks/MISSING\_BLOCK\_ID/actions/disable HTTP/1.1  
HTTP/1.1 404 NotFound

### PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$block = $client->blocks()->disable("BLC123");
```

### Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

block = client.blocks.disable("BLC123")
```

### Ruby

```
@client = GoCardlessPro::Client.new(
    access_token: "your_access_token_here",
    environment: :sandbox
)

block = @client.blocks.disable("BLC123")
```

Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient.newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

Block block = client.blocks().disable("BLC123").execute();
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const block = await client.blocks.disable("BLC456");

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await gocardless.Blocks.DisableAsync("BLC123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }
    block, err := client.Blocks.Disable(context, "BLC123")
}
```

**Enable a block**

Enables a previously disabled block so that it will prevent mandate creation

Relative endpoint: POST /blocks/BLC123/actions/enable

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/blocks/BLC123/actions/enable HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/json
{
    "blocks": {
        "id": "BLC123",
        "block_type": "bank_account",
        "reason_type": "no_intent_to_pay",
        "resource_reference": "BA123",
        "active": "true",
        "created_at": "2021-03-25T17:26:28.305Z",
        "updated_at": "2021-03-25T19:26:28.305Z"
    }
}
```

```
POST https://api.gocardless.com/blocks/BLC123/actions/enable HTTP/1.1
HTTP/1.1 409 Conflict
```

```
POST https://api.gocardless.com/blocks/MISSING_BLOCK_ID/actions/enable HTTP/1.1
HTTP/1.1 404 NotFound
```

## PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$block = $client->blocks()->enable("BLC123");
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
```

```
block = client.blocks.enable("
```

## Ruby

```
@client = GoCardlessPro::Client.new(
    access_token: "your_access_token_here",
    environment: :sandbox
)
block = @client.blocks.enable("
```

## Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient.create(accessToken);
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Consultez notre politique de confidentialité

```
.newBuilder(accessToken)
.withEnvironment(SANDBOX)
.build();

Block block = client.blocks().enable("BLC123").execute();
JavaScript
```

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const block = await client.blocks.enable("BLC456");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);
```

```
var resp = await gocardless.Blocks.EnableAsync("BLC123");
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    block, err := client.Blocks.Enable(context, "BLC123")
}
```

## Create blocks by reference

Creates new blocks for a given reference. By default blocks will be active. Returns 201 if at least one block was created. Returns 200 if there were no new blocks created.

Relative endpoint: POST /blocks/block\_by\_ref

Parameters

**active**  
Shows if the block is active or disabled. Only active blocks will be used when deciding if a mandate should be blocked.  
**reason\_description**  
This field is required if the reason\_type is other. It should be a description of the reason for why you wish to block this payer and why it does not align with the given reason\_types. This is intended to help us improve our knowledge of types of fraud.  
**reason\_type**  
The reason you wish to block this payer, can currently be one of 'identity\_fraud', 'no\_intent\_to\_pay', 'unfair\_chargeback'. If the reason isn't captured by one of the above then 'other' can be selected but you must provide a reason description.  
**reference\_type**  
Type of entity we will seek to get the associated emails and bank accounts to create blocks from. This can currently be one of 'customer' or 'mandate'.  
**reference\_value**  
This field is a reference to the entity you wish to block based on its emails and bank accounts. This may be the ID of a customer or a mandate. This means in order to block by reference the entity must have already been created as a resource.



HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/blocks/block_by_ref HTTP/1.1
```

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

```
{
  "blocks": {
    "reference_type": "customer",
    "reference_value": "CU123",
    "reason_type": "no_intent_to_pay"
  }
}
```

HTTP/1.1 201 Created

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

```
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "blocks": [
    {
      "id": "BLC123",
      "block_type": "email",
      "reason_type": "no_inten",
      "resource_reference": "ex",
      "active": "true",
    }
  ]
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "created_at": "2021-03-25T17:26:28.305Z",
    "updated_at": "2021-03-25T17:26:28.305Z"
  },
  {
    "id": "BLC456",
    "block_type": "bank_account",
    "reason_type": "no_intent_to_pay",
    "resource_reference": "BA123",
    "active": "true",
    "created_at": "2021-03-25T17:26:28.305Z",
    "updated_at": "2021-03-25T17:26:28.305Z"
  }
]
}

```

POST [https://api.gocardless.com/blocks/block\\_by\\_ref](https://api.gocardless.com/blocks/block_by_ref) HTTP/1.1  
Content-Type: application/json

```
{
  "blocks": {
    "reference_type": "customer",
    "reference_value": "CU123",
    "reason_type": "no_intent_to_pay"
  }
}
```

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "blocks": [
    {
      "id": "BLC123",
      "block_type": "email",
      "reason_type": "no_intent_to_pay",
      "resource_reference": "example@example.com",
      "active": "true",
      "created_at": "2021-03-25T17:26:28.305Z",
      "updated_at": "2021-03-25T17:26:28.305Z"
    },
    {
      "id": "BLC456",
      "block_type": "bank_account",
      "reason_type": "no_intent_to_pay",
      "resource_reference": "BA123",
      "active": "true",
      "created_at": "2021-03-25T17:26:28.305Z",
      "updated_at": "2021-03-25T17:26:28.305Z"
    }
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$block = $client->blocks()->blockByRef([
  "params" => [
    "reference_type" => "customer",
    "reference_value" => "CU123",
    "reason_type" => "no_intent_to_pay",
  ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

response = client.blocks.block_by_ref(
  params={
    "reference_type": "customer",
    "reference_value": "CU123",
    "reason_type": "no_intent_to_pay"
  }
)
blocks = response.records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

resp = @client.blocks.block_by_ref(
  params: {
    "reference_type": "customer",
    "reference_value": "CU123",
    "reason_type": "no_intent_to_pay"
  }
)
blocks = resp.records
```

Java

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

Iterable<Block> iterable = client.blocks().blockByRef()
    .withReferenceType("customer")
    .withReferenceValue("CU123")
    .withReasonType("no_intent_to_pay")
    .execute();
List<Block> blocks = Lists.newArrayList(iterable);
JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const response = await client.blocks.block_by_ref(
{
  reference_type: Types.BlockReferenceType.Customer,
  reference_value: 'CU123',
  reason_type: Types.BlockReasonType.NoIntentToPay
}
);
const blocks = response.blocks;

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var resp = await client.Blocks.BlockByRefAsync(new BlockBlockByRefRequest() {
  ReferenceType = "customer",
  ReferenceValue = "CU123",
  ReasonType = "no_intent_to_pay",
});
IReadOnlyList < GoCardless.Resources.Block > blocks = resp.Blocks;

```

Go

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    blockBlockByRefParams := gocardless.BlockBlockByRefParams{
        ReferenceType: "customer",
        ReferenceValue: "CU123",
        ReasonType: "no_intent_to_pay",
    }

    blockBlockByRefResult, err := client.Blocks.BlockByRef(context, blockBlockByRefParams)
    for _, block := range blockBlockByRefResult.Blocks {
        fmt.Println(block.Id)
    }
}

```

## Creditors

Each [payment](#) taken through the API is linked to a “creditor”, to whom the payment is then paid out. In most cases your organisation will have a single “creditor”, but the API also supports collecting payments on behalf of others.

Currently, for Anti Money Laundering reasons, any creditors you add must be directly related to your organisation.

### Properties

**id**  
Unique identifier, beginning with

**address\_line1**  
The first line of the creditor's address

**address\_line2**  
The second line of the creditor's address

**address\_line3**  
The third line of the creditor's address

**bank\_reference\_prefix**  
Prefix for the bank reference sent to that creditor could be ACM

This prefix is also used for:

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

The bank reference of a payout sent



Consultez notre politique de confidentialité

**can\_create\_refunds**  
Boolean indicating whether the creditor is permitted to create refunds.

**city**

The city of the creditor's address.

**country\_code**

[ISO 3166-1 alpha-2 code](#).

**created\_at**

Fixed [timestamp](#), recording when this resource was created.

**creditor\_type**

The type of business of the creditor. Currently, individual, company, charity, partnership, and trust are supported.

**custom\_payment\_pages\_enabled**

Boolean value indicating whether creditor has the [Custom Payment Pages](#) functionality enabled.

**fx\_payout\_currency**

[ISO 4217](#) code for the currency in which amounts will be paid out (after foreign exchange). Currently "AUD", "CAD", "DKK", "EUR", "GBP", "NZD",

"SEK" and "USD" are supported. Present only if payouts will be (or were) made via foreign exchange.

**logo\_url**

URL for the creditor's logo, which may be shown on their payment pages.

**mandate\_imports\_enabled**

Boolean value indicating whether creditor has the [Mandate Imports](#) functionality enabled.

**merchant\_responsible\_for\_notifications**

Boolean value indicating whether the organisation is responsible for sending all customer notifications (note this is separate from the functionality described [here](#)). If you are a partner app, and this value is true, you should not send notifications on behalf of this organisation.

**name**

The creditor's trading name.

**postal\_code**

The creditor's postal code.

**region**

The creditor's address region, county or department.

**scheme\_identifiers**

An array of the scheme identifiers this creditor can create mandates against.

The support address, `phone_number` and `email` fields are for customers to contact the merchant for support purposes. They must be displayed on the payment page, please see our [compliance requirements](#) for more details.

Each instance will contain these properties:

- `address_line1`: The first line of the scheme identifier's support address.
- `address_line2`: The second line of the scheme identifier's support address.
- `address_line3`: The third line of the scheme identifier's support address.
- `can_specify_mandate_reference`: Whether a custom reference can be submitted for mandates using this scheme identifier.
- `city`: The city of the scheme identifier's support address.
- `country_code`: [ISO 3166-1 alpha-2 code](#).
- `created_at`: Fixed [timestamp](#), recording when this resource was created.
- `currency`: The currency of the scheme identifier.
- `email`: Scheme identifier's support email address.
- `id`: Unique identifier, usually beginning with "SU".
- `minimum_advance_notice`: The minimum interval, in working days, between the sending of a pre-notification to the customer, and the charge date of a payment using this scheme identifier.

By default, GoCardless sends these notifications automatically. Please see our [compliance requirements](#) for more details.

- `name`: The name which appears on customers' bank statements. This should usually be the merchant's trading name.
- `phone_number`: Scheme identifier's support phone number.
- `postal_code`: The scheme identifier's support postal code.
- `reference`: The scheme-unique identifier against which payments are submitted.
- `region`: The scheme identifier's support address region, county or department.
- `scheme`: The scheme which this scheme identifier applies to.
- `status`: The status of the scheme identifier. Only active scheme identifiers will be applied to a creditor and used against payments.

**verification\_status**

The creditor's verification status, indicating whether they can yet receive payouts. For more details on handling verification as a partner, see our ["Helping your users get verified" guide](#). O

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

verified, they may in the future the new bank account before

; before they can be verified and

and should visit the verification

`links[default_aud_payout_account]`  
ID of the [bank account](#) which  
`links[default_cad_payout_account]`



Consultez notre politique de confidentialité

ID of the [bank account](#) which is set up to receive payouts in CAD.  
links[default\_dkk\_payout\_account]  
ID of the [bank account](#) which is set up to receive payouts in DKK.  
links[default\_eur\_payout\_account]  
ID of the [bank account](#) which is set up to receive payouts in EUR.  
links[default\_gbp\_payout\_account]  
ID of the [bank account](#) which is set up to receive payouts in GBP.  
links[default\_nzd\_payout\_account]  
ID of the [bank account](#) which is set up to receive payouts in NZD.  
links[default\_sek\_payout\_account]  
ID of the [bank account](#) which is set up to receive payouts in SEK.  
links[default\_usd\_payout\_account]  
ID of the [bank account](#) which is set up to receive payouts in USD.

## Create a creditor

Creates a new creditor.

Relative endpoint: POST /creditors

**Restricted:** This endpoint is restricted to GoCardless Embed customers. Please [contact us](#) if you are interested in using this product. Partners should instead manage multiple merchant accounts by building a [partner integration](#).

## Parameters

**country\_code**  
*required ISO 3166-1 alpha-2 code*

required ISO 5100-1 alpha-2 code.  
**creditor\_type**

*required* The type of business of the company

**bank\_reference\_prefix** Prefix for the bank reference of payouts sent to this creditor. For instance, if the creditor's bank\_reference\_prefix was ACME, the bank reference of a payout sent to that creditor could be ACME\_88789.

This prefix is also used for refunds in EUR and GBP.

• • • • • • •

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com/creditors HTTP/1.1

Content-Type: application/json

```
{  
  "creditors": {  
    "name": "Acme",  
    "country_code": "GB",  
    "creditor_type": "company",  
    "bank_reference_prefix": "ACME"  
  }  
}
```

HTTP/1.1 201 Created  
Location: /creditors/CR123

Content-Type: application/json

```
{  
  "creditions": {
```

```
    "creditors": {
      "id": "CR123",
      "created_at": "2017-02-16T12:34:56.123Z",
      "name": "Acme",
      "address_line1": null,
      "address_line2": null,
      "address_line3": null,
      "city": null,
      "region": null,
      "postal_code": null,
      "country_code": "GB",
      "creditor_type": "company",
      "logo_url": null,
      "scheme_identifiers": [],
      "verification_status": "successful",
      "can_create_refunds": false,
      "fx_payout_currency": null,
      "mandate_imports_enabled": false,
      "custom_payment_pages_enabled": true,
      "merchant_responsible_for_notifications": true,
      "bank_reference_prefix": "ACME",
      "links": {}
    }
  }
```

BUD

```
$client = new \GoCardlessPro\Client();
    'access_token' => 'your_access_token',
    'environment'  => \GoCardlessPro\Environment::TEST);
```

]);

# Python

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.creditors.create(params={
    "name": "Acme",
    "country_code": "GB",
    "creditor_type": "company",
    "bank_reference_prefix": "ACME",
})
Ruby
```

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.creditors.create(
  params: {
    name: "Acme",
    country_code: "GB",
    creditor_type: "company",
    bank_reference_prefix: "ACME",
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();
```

```
Creditor creditor = client.creditors().create()
    .withName("Acme")
    .withCountryCode("GB")
    .withCreditorType("company")
    .withBankReferencePrefix("ACME")
    .execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditor = await client.creditors.create({
  name: "Acme",
  country_code: "GB",
  creditor_type: "company",
  bank_reference_prefix: "ACME",
});
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorRequest = new GoCardless.Services.CreditorCreateRequest()
{
    Name = "Acme",
    CountryCode = "GB",
    CreditorType = "company",
    BankReferencePrefix = "ACME",
};

var creditorResponse = await gocardless.Creditors.CreateAsync(creditorRequest);
GoCardless.Resources.Creditor creditor = creditorResponse.Creditor;
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in init")
        return
    }

    creditorCreateParams := goca
    Name:          "Acme"
    CountryCode:   "GB",
    CreditorType:  "comp
    BankReferencePrefix: "ACME
}

    creditor, err := client.Cred
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

List creditors

Returns a [cursor-paginated](#) list of your creditors.

Relative endpoint: GET /creditors

#### Parameters

**after** Cursor pointing to the start of the desired set.

**before** Cursor pointing to the end of the desired set.

**created\_at[gt]** Limit to records created after the specified date-time.

**created\_at[gte]** Limit to records created on or after the specified date-time.

**created\_at[lt]** Limit to records created before the specified date-time.

**created\_at[lte]** Limit to records created on or before the specified date-time.

**limit** Number of records to return.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET <https://api.gocardless.com/creditors> HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null,
    },
    "limit": 50
  },
  "creditors": [
    {
      "id": "CR123",
      "created_at": "2017-02-16T12:34:56.000Z",
      "name": "Acme",
      "address_line1": null,
      "address_line2": null,
      "address_line3": null,
      "city": null,
      "region": null,
      "postal_code": null,
      "country_code": "GB",
      "creditor_type": "company",
      "logo_url": null,
      "scheme_identifiers": [],
      "verification_status": "successful",
      "can_create_refunds": false,
      "fx_payout_currency": null,
      "mandate_imports_enabled": false,
      "custom_payment_pages_enabled": true,
      "merchant_responsible_for_notifications": true,
      "bank_reference_prefix": "ACME",
      "links": {}
    }
  ]
}
```

#### PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->creditors()->list();

$client->creditors()->list([
  'params' => ['created_at[gt]' => '2014-05-08T17:01:06.000Z']
])
```

#### Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.creditors.list().records
```

#### Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token_here",
  environment: :sandbox
)
```

```
@client.creditors.list
```

```
@client.creditors.list(params: {
```

#### Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

for (Creditor creditor : client.creditors().all().execute()) {
    System.out.println(creditor.getId());
}
JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditors = await client.creditors.list();

// List the first three creditors.
const creditors = await client.creditors.list({ limit: 3 });

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorListResponse = gocardless.Creditors.All();
foreach (GoCardless.Resources.Creditor creditor in creditorListResponse)
{
    Console.WriteLine(creditor.Name);
}

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    creditorListParams := gocardless.CreditorListParams{
        Limit: 3,
    }

    // List the first three creditors.
    creditorListResult, err := client.Creditors.List(context, creditorListParams)
    for _, creditor := creditorListResult.Creditors {
        fmt.Println(creditor.Name)
    }
}

```

## Get a single creditor

Retrieves the details of an existing creditor.

Relative endpoint: GET /creditors/CR123

Parameters



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/creditors/CR123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "creditors": {
        "id": "CR123",
        "created_at": "2017-02-16T12:34:56.000Z",
        "name": "Acme",
        "address_line1": null,
        "address_line2": null,
        "address_line3": null,
        "city": null,
        "region": null,
        "postal_code": null,
        "country_code": "GB",
        "creditor_type": "company",
        "logo_url": null,
        "scheme_identifiers": [],
        "verification_status": "su",
        "can_create_refunds": false,
        "fx_payout_currency": null,
        "mandate_imports_enabled": false,
        "custom_payment_pages_enabled": false,
        "merchant_responsible_for_fees": false,
        "bank_reference_prefix": null,
        "links": {}
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->creditors()->get('CR123');
Python
```

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.creditors.get("CR123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.creditors.get("CR123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

Creditor creditor = client.creditors().get("CR123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditor = await client.creditor.find("CR123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorResponse = await gocardless.Creditors.GetAsync("CR0123");
GoCardless.Resources.Creditor creditor = creditorResponse.Creditor;
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    creditor, err := client.Creditors.Get(context, "CR123")
}
```

## Update a creditor

Updates a creditor object. Supports all of the fields supported when creating a creditor.

Relative endpoint: PUT /creditors/CR123

Parameters

`address_line1`  
The first line of the creditor's address.

`address_line2`  
The second line of the creditor's address.

`address_line3`  
The third line of the creditor's address.

`bank_reference_prefix`  
Prefix for the bank reference sent to that creditor could be ACM.

`city`  
This prefix is also used for:

`country_code`  
[ISO 3166-1 alpha-2 code](#).

`name`

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

the bank reference of a payout sent

The creditor's trading name.

`postal_code`

The creditor's postal code.

`region`

The creditor's address region, county or department.

`links[default_aud_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in AUD.

`links[default_cad_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in CAD.

`links[default_dkk_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in DKK.

`links[default_eur_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in EUR.

`links[default_gbp_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in GBP.

`links[default_nzd_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in NZD.

`links[default_sek_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in SEK.

`links[default_usd_payout_account]`

ID of the [bank account](#) which is set up to receive payouts in USD.

● ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

PUT <https://api.gocardless.com/creditors/CR123> HTTP/1.1

Content-Type: application/json

```
{
  "creditors": {
    "links": {
      "default_gbp_payout_account": "BA789"
    }
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "creditors": {
    "id": "CR123",
    "created_at": "2017-02-16T12:34:56.000Z",
    "name": "Acme",
    "address_line1": null,
    "address_line2": null,
    "address_line3": null,
    "city": null,
    "region": null,
    "postal_code": null,
    "country_code": "GB",
    "creditor_type": "company",
    "logo_url": null,
    "scheme_identifiers": [],
    "verification_status": "successful",
    "can_create_refunds": false,
    "fx_payout_currency": null,
    "mandate_imports_enabled": false,
    "custom_payment_pages_enabled": true,
    "merchant_responsible_for_notifications": true,
    "bank_reference_prefix": "ACME",
    "links": {
      "default_gbp_payout_account": "BA789",
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->creditors()->update('CR123', [
  'params' => ['links' => ['default_gbp_payout_account' => 'BA789']]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
```

```
client.creditors.update("CR123", params={
  "links": {
    "default_gbp_payout_account": "BA789"
  }
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token_here",
  environment: :sandbox
)

@client.creditors.update(
  "CR123",
  params: {
    links: { default_gbp_payout_account: "BA789" }
  }
)
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.creditors().update("CR123")
    .withLinksDefaultGbpPayoutAccount("BA789")
    .execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditor = await client.creditors.update(
  "CR123",
  {
    links: {
      default_gbp_payout_account: "BA789"
    }
  }
);
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorRequest = new GoCardless.Services.CreditorUpdateRequest()
{
    Links = new GoCardless.Services.CreditorUpdateRequest.CreditorLinks()
    {
        DefaultGbpPayoutAccount = "BA789"
    }
};

var creditorResponse = await gocardless.Creditors.UpdateAsync("CR0123", creditorRequest);
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    creditorUpdateParams := gocardless.CreditorUpdateParams{
        Links: &gocardless.CreditorUpdateParamsLinks{
            DefaultGbpPayoutAccount: "BA789",
        },
    }

    creditor, err := client.Creditors.Update(context, "CR123", creditorUpdateParams)
}
```

## Creditor Bank Accounts

Creditor Bank Accounts hold the bank details of a [creditor](#). These are the bank accounts which your [payouts](#) will be sent to.

Note that creditor bank accounts must be unique, and so you will encounter a `bank_account_exists` error if you try to create a duplicate bank account. You may wish to handle this by updating the existing record instead, the ID of which will be provided as `links[creditor_bank_account]` in the error response.

**Restricted:** This API is not available for partner integrations.

## Properties

<code>id</code>	Unique identifier, beginning
<code>account_holder_name</code>	Name of the account holder upcased and truncated to 18
<code>account_numberEnding</code>	The last few digits of the ac
<code>account_type</code>	Bank account type. Require information.
<code>bank_name</code>	Name of bank, taken from t
<code>country_code</code>	<a href="#">ISO 3166-1 alpha-2 code</a> . Defaults to the country code of the iban if supplied, otherwise is required.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

ield will be transliterated,

s. See [local details](#) for more

 Consultez notre politique de confidentialité



```
"params" => ["account_number" => "55779911",
  "branch_code" => "200000",
  "country_code" => "GB",
  "account_holder_name" => "Acme",
  "links" => ["creditor" => "CR123"]]
});
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.creditor_bank_accounts.create(params={
  "account_number": "55779911",
  "branch_code": "200000",
  "country_code": "GB",
  "account_holder_name": "Acme",
  "links": {
    "creditor": "CR123"
  }
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.creditor_bank_accounts.create(
  params: {
    account_holder_name: "Example Ltd",
    account_number: "55779911",
    branch_code: "200000",
    country_code: "GB",
    links: {
      creditor: "CR123"
    }
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

CreditorBankAccount creditorBankAccount = client.creditorBankAccounts().create()
  .withAccountNumber("55779911")
  .withBranchCode("200000")
  .withCountryCode("GB")
  .withAccountHolderName("Acme")
  .withLinksCreditor("CR123")
  .execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditorBankAccount = await client.creditorBankAccounts.create({
  account_number: "55779911",
  branch_code: "200000",
  country_code: "GB",
  account_holder_name: "Acme",
  links: {
    creditor: "CR123"
  }
});
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorBankAccountRequest = new GoCardless.Services.CreditorBankAccountCreateRequest()
{
  AccountHolderName = "Example Ltd",
  AccountNumber = "55779911",
  BranchCode = "200000",
  CountryCode = "GB",
  Links = new GoCardless.Services.CreditorBankAccountCreateRequest.CreditorBankAccountLinks()
  {
    Creditor = "CR0123"
  }
};
```

```
var creditorBankAccountResponse = gocardless.CreditorBankAccounts.Create(creditorBankAccountRequest);
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    fmt.Printf("got err in initialising config: %s", err.Error())
    return
}
client, err := gocardless.New(config)
if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
}

creditorBankAccountCreateParams := gocardless.CreditorBankAccountCreateParams{
    AccountNumber:          "55779911",
    BranchCode:             "200000",
    CountryCode:            "GB",
    AccountHolderName:      "Acme",
}
creditorBankAccount, err := client.CreditorBankAccounts.Create(context, creditorBankAccountCreateParams)
}

```

## List creditor bank accounts

Returns a [cursor-paginated](#) list of your creditor bank accounts.

Relative endpoint: GET /creditor\_bank\_accounts

### Parameters

`after`  
Cursor pointing to the start of the desired set.  
`before`  
Cursor pointing to the end of the desired set.  
`created_at[gt]`  
Limit to records created after the specified date-time.  
`created_at[gte]`  
Limit to records created on or after the specified date-time.  
`created_at[lt]`  
Limit to records created before the specified date-time.  
`created_at[lte]`  
Limit to records created on or before the specified date-time.  
`creditor`  
Unique identifier, beginning with “CR”.  
`enabled`  
If true, only return enabled bank accounts. If false, only return disabled bank accounts.  
`limit`  
Number of records to return.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/creditor\_bank\_accounts HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "meta": {
        "cursors": {
            "before": null,
            "after": "BA123",
        },
        "limit": 50
    },
    "creditor_bank_accounts": [
        {
            "id": "BA123",
            "created_at": "2014-05-27T12:43:17.000Z",
            "account_holder_name": "Acme",
            "account_numberEnding": "11",
            "account_type": null,
            "country_code": "GB",
            "currency": "GBP",
            "bank_name": "BARCLAYS BANK PLC",
            "enabled": true,
            "links": {
                "creditor": "CR123"
            },
            "verification_status": "pending"
        },
        ...
    ]
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment::TEST;
]);
$client->creditorBankAccounts([
    $client->creditorBankAccounts([
        $client->creditorBankAccounts([
            'params' => ['enabled' => true]
        ]);
    ]);
]);
```

Python

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.creditor_bank_accounts.list().records

client.creditor_bank_accounts.list(params={"creditor": "CR123"}).records
Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
@client.creditor_bank_accounts.list
@client.creditor_bank_accounts.list(params: { enabled: true })

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

for (CreditorBankAccount creditorBankAccount : client.creditorBankAccounts().all().withCreditor("CR123").execute()) {
    System.out.println(creditorBankAccount.getId());
}

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditorBankAccounts = await client.creditorBankAccounts.list();

// List all 'enabled' creditor bank accounts.
const creditorBankAccounts = await client.creditorBankAccounts.list({ enabled: true });

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorBankAccountRequest = new GoCardless.Services.CreditorBankAccountListRequest()
{
    Creditor = "CR000123"
};

var creditorBankAccountListResponse = gocardless.CreditorBankAccounts.All(creditorBankAccountRequest);
foreach (GoCardless.Resources.CreditorBankAccount creditorBankAccount in creditorBankAccountListResponse)
{
    Console.WriteLine(creditorBankAccount.Id);
}

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    creditorBankAccountListParams := gocardless.CreditorBankAccountListParams{}
    creditorBankAccountListResult, err := client.CreditorBankAccounts.List(context, creditorBankAccountListParams)
    for _, creditorBankAccount := range creditorBankAccountListResult.CreditorBankAccounts {
        fmt.Println(creditorBankAccount.Id)
    }
}
```

**Get a single creditor bank ac-**

Retrieves the details of an existing

Relative endpoint: GET /creditor

- HTTP
  - PHP
  - Python
  - Ruby
  - Java
  - JavaScript
  - C/C++

GET <https://api.gocardless.com>

```
HTTP/1.1 200 OK
Content-Type: application/json
{
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "id": "BA123",
    "created_at": "2014-05-27T12:43:17.000Z",
    "account_holder_name": "Acme",
    "account_numberEnding": "11",
    "account_type": null,
    "country_code": "GB",
    "currency": "GBP",
    "bank_name": "BARCLAYS BANK PLC",
    "enabled": true,
    "links": {
      "creditor": "CR123"
    },
    "verification_status": "pending"
  }
}

```

PHP

```

$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

```

```
$client->creditorBankAccounts()->get("BA123");
```

Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.creditor_bank_accounts.get("BA123")

```

Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

```

```
@client.creditor_bank_accounts.get("BA123")
```

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

CreditorBankAccount creditorBankAccount = client.creditorBankAccounts().get("BA123").execute();

```

JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditorBankAccount = await client.creditorBankAccounts.find('BA123');

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorBankAccountResponse = await gocardless.CreditorBankAccounts.GetAsync("BA0123");
GoCardless.Resources.CreditorBankAccount creditorBankAccount = creditorBankAccountResponse.CreditorBankAccount;

```

Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  creditorBankAccount, err := client.CreditorBankAccounts.Get("BA123")
}

```

## Disable a creditor bank account

Immediately disables the bank account.

This will return a `disable_failed` response.

A disabled bank account can be re-enabled.

Relative endpoint: POST /creditorBankAccounts/{id}/disable

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/creditor_bank_accounts/BA123/actions/disable HTTP/1.1
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "creditor_bank_accounts": {
    "id": "BA123",
    "created_at": "2014-05-27T12:43:17.000Z",
    "account_holder_name": "Acme",
    "account_numberEnding": "11",
    "account_type": null,
    "country_code": "GB",
    "currency": "GBP",
    "bank_name": "BARCLAYS BANK PLC",
    "enabled": false,
    "links": {
      "creditor": "CR123"
    },
    "verification_status": "pending"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->creditorBankAccounts()->disable("BA123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.creditor_bank_accounts.disable("BA123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.creditor_bank_accounts.disable("BA123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.creditorBankAccounts().disable("BA123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const creditorBankAccountResponse = await client.creditorBankAccounts.disable("BA123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var creditorBankAccountResponse = await gocardless.CreditorBankAccounts.DisableAsync("BA0123");
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in init: %v", err)
    return
  }
  creditorBankAccount, err := client.CreditorBankAccounts.Disable("BA123")
  if err != nil {
    fmt.Println("error in disable: %v", err)
  }
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

[Validate Creditor Bank Account](#)

Validate bank details without creating a creditor bank account

Relative endpoint: POST /creditor\_bank\_accounts/validate

Parameters

`iban`

International Bank Account Number. Alternatively you can provide [local\\_details](#). IBANs are not accepted for Swedish bank accounts denominated in SEK - you must supply [local\\_details](#).

`local_details[bank_number]`

Bank account number - see [local\\_details](#) for more information. Alternatively you can provide an `iban`.

`local_details[sort_code]`

Branch code - see [local\\_details](#) for more information. Alternatively you can provide an `iban`.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

PHP

Python

Ruby

Java

JavaScript

.NET

Go

## Currency Exchange Rates

Currency exchange rates from our foreign exchange provider.

Properties

`rate`

The exchange rate from the source to target currencies provided with up to 10 decimal places.

`source`

Source currency

`target`

Target currency

`time`

Time at which the rate was retrieved from the provider.

### List exchange rates

Returns a [cursor-paginated](#) list of all exchange rates.

Relative endpoint: GET /currency\_exchange\_rates

Parameters

`after`

Cursor pointing to the start of the desired set.

`before`

Cursor pointing to the end of the desired set.

`limit`

Number of records to return.

`source`

Source currency

`target`

Target currency

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

GET [https://api.gocardless.com/currency\\_exchange\\_rates](https://api.gocardless.com/currency_exchange_rates)

HTTP/1.1 200 OK

Content-Type: application/json

{

```
  "currency_exchange_rates": [
    {
      "rate": "1.19111",
      "source": "GBP",
    }
  ]
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "target": "EUR",
        "time": "2020-02-24T23:50:04.000Z"
    }
],
"meta": {
    "cursors": {
        "after": null,
        "before": null
    },
    "limit": 50
}
}
PHP

```

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->currencyExchangeRates()->list([
    'params' => ["source" => "EUR", "target" => "GBP"]
]);

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.currency_exchange_rates.list(params={"source": "EUR", "target": "GBP"}).records

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.currency_exchange_rates.list(params: { source: "EUR", target: "GBP" })

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

```

```
client.currencyExchangeRates().all().withSource("EUR").withTarget("GBP").execute()
```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const currencyExchangeRates = await client.currencyExchangeRates.list();

// List the currency exchange rate for a source and target currency.
const currencyExchangeRates = await client.currencyExchangeRates.list({ source: 'EUR', target: 'GBP' });

```

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var request = new GoCardless.Services.CurrencyExchangeRateListRequest()
{
    Source = "EUR",
    Target = "GBP"
};

var response = gocardless.CurrencyExchangeRates.All(request);
foreach (GoCardless.Resources.CurrencyExchangeRate rate in response)
{
    Console.WriteLine(rate.Rate);
}

```

## Go

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_t
    config, err := gocardless.NewConfig()
    if err != nil {
        fmt.Printf("got err in ini
        return
    }
    client, err := gocardless.NewClient(config)
    if err != nil {
        fmt.Println("error in init
        return
    }

    currencyExchangeRateListParams := &GoCardless.Resources.CurrencyExchangeRateListParams{
        Source: "EUR",
        Target: "GBP",
    }

    currencyExchangeRateListResult, err := client.CurrencyExchangeRates.List(context, currencyExchangeRateListParams)
}

```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

for _, currencyExchangeRate := range currencyExchangeRateListResult.CurrencyExchangeRates {
    fmt.Println(currencyExchangeRate.Id)
}
}

```

## Customers

Customer objects hold the contact details for a customer. A customer can have several [customer bank accounts](#), which in turn can have several Direct Debit [mandates](#).

### Properties

**id**  
Unique identifier, beginning with “CU”.

**address\_line1**  
The first line of the customer’s address.

**address\_line2**  
The second line of the customer’s address.

**address\_line3**  
The third line of the customer’s address.

**city**  
The city of the customer’s address.

**company\_name**  
Customer’s company name. Required unless a `given_name` and `family_name` are provided. For Canadian customers, the use of a `company_name` value will mean that any mandate created from this customer will be considered to be a “Business PAD” (otherwise, any mandate will be considered to be a “Personal PAD”).

**country\_code**  
[ISO 3166-1 alpha-2 code](#).

**created\_at**  
Fixed [timestamp](#), recording when this resource was created.

**danish\_identity\_number**  
For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer’s bank account is denominated in Danish krone (DKK).

**email**  
Customer’s email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

**family\_name**  
Customer’s surname. Required unless a `company_name` is provided.

**given\_name**  
Customer’s first name. Required unless a `company_name` is provided.

**language**  
[ISO 639-1](#) code. Used as the language for notification emails sent by GoCardless if your organisation does not send its own (see [compliance requirements](#)). Currently only “en”, “fr”, “de”, “pt”, “es”, “it”, “nl”, “da”, “nb”, “sl”, “sv” are supported. If this is not provided, the language will be chosen based on the `country_code` (if supplied) or default to “en”.

**metadata**  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**phone\_number**  
[ITU E.123](#) formatted phone number, including country code.

**postal\_code**  
The customer’s postal code.

**region**  
The customer’s address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

**swedish\_identity\_number**  
For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer’s bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.

## Create a customer

Creates a new customer object.

Relative endpoint: POST /customers

**Restricted:** this endpoint is [restricted](#) to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with approved payment pages.

**Warning:** this endpoint is a legacy API endpoint and does not support GoCardless’ newest features. We recommend using the [Billing Requests API](#) instead.

### Parameters

**address\_line1**  
The first line of the customer’s address.

**address\_line2**  
The second line of the customer’s address.

**address\_line3**  
The third line of the customer’s address.

**city**  
The city of the customer’s address.

**company\_name**  
Customer’s company name. Required unless a `given_name` and `family_name` are provided. For Canadian customers, the use of a `company_name` value will mean that any mandate created from this customer will be considered to be a “Business PAD” (otherwise, any mandate will be considered to be a “Personal PAD”).

**country\_code**  
[ISO 3166-1 alpha-2 code](#).

**danish\_identity\_number**  
For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer’s bank account is denominated in Danish krone (DKK).

**email**  
Customer’s email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

**family\_name**  
Customer’s surname. Required unless a `company_name` is provided.

**given\_name**

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

a `company_name` value will mean that any mandate created from this customer will be considered to be a “Business PAD” (otherwise, any mandate will be considered to be a “Personal PAD”).

nk account is denominated in

Consultez notre politique de confidentialité



Customer's first name. Required unless a `company_name` is provided.

#### language

[ISO 639-1](#) code. Used as the language for notification emails sent by GoCardless if your organisation does not send its own (see [compliance requirements](#)).

Currently only "en", "fr", "de", "pt", "es", "it", "nl", "da", "nb", "sl", "sv" are supported. If this is not provided, the language will be chosen based on the `country_code` (if supplied) or default to "en".

#### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### phone\_number

[ITU E.123](#) formatted phone number, including country code.

#### postal\_code

The customer's postal code.

#### region

The customer's address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

#### swedish\_identity\_number

For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer's bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.



HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

POST <https://api.gocardless.com/customers> HTTP/1.1

Content-Type: application/json

```
{
  "customers": {
    "email": "user@example.com",
    "given_name": "Frank",
    "family_name": "Osborne",
    "address_line1": "27 Acer Road",
    "address_line2": "Apt 2",
    "city": "London",
    "postal_code": "E8 3GX",
    "country_code": "GB",
    "metadata": {
      "salesforce_id": "ABCD1234"
    }
  }
}
```

HTTP/1.1 201 Created

Location: /customers/CU123

Content-Type: application/json

```
{
  "customers": {
    "id": "CU123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "email": "user@example.com",
    "given_name": "Frank",
    "family_name": "Osborne",
    "address_line1": "27 Acer Road",
    "address_line2": "Apt 2",
    "address_line3": null,
    "city": "London",
    "region": null,
    "postal_code": "E8 3GX",
    "country_code": "GB",
    "language": "en",
    "phone_number": null,
    "swedish_identity_number": null,
    "danish_identity_number": null,
    "metadata": {
      "salesforce_id": "ABCD1234"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->customers()->create([
  'params' => ['email' => 'tim@gocardless.com',
    'given_name' => "Tim",
    'family_name' => "Rogers",
    'country_code' => "GB"]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.customers.create(params:
  'email': "user@example.com",
  'given_name': "Frank",
  'family_name': "Osborne",
  'address_line1': "27 Acer Ro",
  'address_line2': "Apt 2",
  'city': "London",
  'postal_code': "E8 3GX",
  'country_code': "GB",
  'metadata': {
    "salesforce_id": "ABCD1234"
  }
)
```

Ruby

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.customers.create(
  params: {
    email: "user@example.com",
    given_name: "Jacob",
    family_name: "Pargin",
    country_code: "GB"
  }
)
Java
```

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
Customer customer = client.customers().create()
  .withEmail("user@example.com")
  .withGivenName("Frank")
  .withFamilyName("Osborne")
  .withAddressLine1("27 Acer Road")
  .withAddressLine2("Apt 2")
  .withCity("London")
  .withPostalCode("E8 3GX")
  .withCountryCode("GB")
  .withMetadata("salesforce_id", "ABCD1234")
  .execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customer = await client.customers.create({
  email: "user@example.com",
  given_name: "Frank",
  family_name: "Osborne",
  address_line1: "27 Acer Road",
  address_line2: "Apt 2",
  city: "London",
  postal_code: "E8 3GX",
  country_code: "GB",
  metadata: {
    salesforce_id: "ABCD1234"
  }
});
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerRequest = new GoCardless.Services.CustomerCreateRequest()
{
  Email = "user@example.com",
  GivenName = "Frank",
  FamilyName = "Osborne",
  AddressLine1 = "27 Acer Road",
  AddressLine2 = "Apt 2",
  City = "London",
  PostalCode = "E8 3GX",
  CountryCode = "GB",
  Metadata = new Dictionary<string, string>()
  {
    {"salesforce_id", "ABCD1234"}
  }
};

var customerResponse = await gocardless.Customers.CreateAsync(customerRequest);
GoCardless.Resources.Customer customer = customerResponse.Customer;
```

## Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_t
  config, err := gocardless.New
  if err != nil {
    fmt.Printf("got err in ini
    return
  }
  client, err := gocardless.New
  if err != nil {
    fmt.Println("error in init
    return
  }

  customerCreateParams := gocar
  AddressLine1: "27 Acer Roa
  AddressLine2: "Apt 2",
  City: "London",
  PostalCode: "E8 3GX",
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    CountryCode: "GB",
    Email: "user@example.com",
    GivenName: "Frank",
    FamilyName: "Osborne",
}

customer, err := client.Customers.Create(context, customerCreateParams)
}

```

## List customers

Returns a [cursor-paginated](#) list of your customers.

Relative endpoint: GET /customers

### Parameters

**after**  
Cursor pointing to the start of the desired set.  
**before**  
Cursor pointing to the end of the desired set.  
**created\_at[gt]**  
Limit to records created after the specified date-time.  
**created\_at[gte]**  
Limit to records created on or after the specified date-time.  
**created\_at[lte]**  
Limit to records created before the specified date-time.  
**created\_at[lt]**  
Limit to records created on or before the specified date-time.  
**currency**  
[ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.  
**limit**  
Number of records to return.

**sort\_direction**  
The direction to sort in. One of:

- asc
- desc

**sort\_field**  
Field by which to sort records. One of:

- name
- company\_name
- created\_at

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/customers?after=CU123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "before": "CU000",
      "after": "CU456",
    },
    "limit": 50
  },
  "customers": [
    {
      "id": "CU123",
      "created_at": "2014-05-08T17:01:06.000Z",
      "email": "user@example.com",
      "given_name": "Frank",
      "family_name": "Osborne",
      "address_line1": "27 Acer Road",
      "address_line2": "Apt 2",
      "address_line3": null,
      "city": "London",
      "region": null,
      "postal_code": "E8 3GX",
      "country_code": "GB",
      "language": "en",
      "metadata": {
        "salesforce_id": "ABCD123"
      }
    },
    ...
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Customers();
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment::TEST;
]);
$cclient->customers()->list();
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
$client->customers()->list([
    "params" => ["created_at[gt]" => "2015-11-03T09:30:00Z"]
]);
Python

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.customers.list().records
client.customers.list(params={"after": "CU123"}).records

Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.customers.list

@client.customers.list(
  params: {
    "created_at[gt]" => "2016-08-06T09:30:00Z"
  }
)

@client.customers.list.records.each { |customer| puts customer.inspect }

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

for (Customer customer : client.customers().all().execute()) {
    System.out.println(customer.getId());
}

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customer = await client.customers.list();

// List customers with a given currency.
await client.customers.list({ currency: "GBP" });

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerListResponse = gocardless.Customers.All();
foreach (GoCardless.Resources.Customer customer in customerListResponse)
{
    Console.WriteLine(customer.GivenName);
}

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    customerListParams := gocardless.CustomerListParams{
        Currency: "GBP",
    }

    customerListResult, err := c.
    for _, customer GivenName ra
        fmt.Println(customer.Giv
    }
}

Get a single customer

Retrieves the details of an existing
Relative endpoint: GET /customers/
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/customers/CU123 HTTP/1.1

```
HTTP/1.1 200 OK
Content-Type: application/json
{
```

```
{  
  "customers": {  
    "id": "CU123",  
    "created_at": "2014-05-08T17:01:06.000Z",  
    "email": "user@example.com",  
    "given_name": "Frank",  
    "family_name": "Osborne",  
    "address_line1": "27 Acer Road",  
    "address_line2": "Apt 2",  
    "address_line3": null,  
    "city": "London",  
    "region": null,  
    "postal_code": "E8 3GX",  
    "country_code": "GB",  
    "language": "en",  
    "phone_number": null,  
    "swedish_identity_number": null,  
    "danish_identity_number": null,  
    "metadata": {  
      "salesforce_id": "ABCD1234"  
    }  
  }  
}
```

GET https://api.gocardless.com/customers/CU123 HTTP/1.1

```
HTTP/1.1 410 Gone
Content-Type: application/json
{
  "error": {
    "documentation_url": "https://developer.gocardless.com/api-reference#customer_data_removed",
    "message": "This customer data has been removed",
    "type": "invalid_api_usage",
    "errors": [
      {
        "reason": "customer_removed",
        "message": "This customer data has been removed"
      }
    ],
    "code": 410,
    "request_id": "deadbeef-0000-4000-0000-444400004444"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->customers()->get("CU123");
```

# Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.customers.get("CU123")
```

Ruby

```
@client = GoCardlessPro::Client.new(  
  access_token: "your_access_token",  
  environment: :sandbox  
)
```

```
@client.customers.get("CU123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();
```

```
Customer customer = client.cust
```

## JavaScript

```
const constants = require('gocardless-nodejs').constants;
const gocardless = require('gocardless-nodejs');
const client = gocardless('your-client-id');
```

```
const customer = await client.
```

```
String accessToken = "your_acce  
GoCardlessClient gocardless =  
  
var customerResponse = await g
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de



Consultez notre politique de confidentialité

```

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    customer, err := client.Customers.Get(context, "CU123")
}

```

## Update a customer

Updates a customer object. Supports all of the fields supported when creating a customer.

Relative endpoint: PUT /customers/CU123

Parameters

`address_line1`  
The first line of the customer's address.

`address_line2`  
The second line of the customer's address.

`address_line3`  
The third line of the customer's address.

`city`  
The city of the customer's address.

`company_name`  
Customer's company name. Required unless a `given_name` and `family_name` are provided. For Canadian customers, the use of a `company_name` value will mean that any mandate created from this customer will be considered to be a "Business PAD" (otherwise, any mandate will be considered to be a "Personal PAD").

`country_code`  
[ISO 3166-1 alpha-2 code](#).

`danish_identity_number`  
For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer's bank account is denominated in Danish krone (DKK).

`email`  
Customer's email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

`family_name`  
Customer's surname. Required unless a `company_name` is provided.

`given_name`  
Customer's first name. Required unless a `company_name` is provided.

`language`  
[ISO 639-1](#) code. Used as the language for notification emails sent by GoCardless if your organisation does not send its own (see [compliance requirements](#)). Currently only "en", "fr", "de", "pt", "es", "it", "nl", "da", "nb", "sl", "sv" are supported. If this is not provided, the language will be chosen based on the `country_code` (if supplied) or default to "en".

`metadata`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`phone_number`  
[ITU E.123](#) formatted phone number, including country code.

`postal_code`  
The customer's postal code.

`region`  
The customer's address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

`swedish_identity_number`  
For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer's bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

PUT <https://api.gocardless.com/customers/CU123> HTTP/1.1

Content-Type: application/json

```
{
    "customers": {
        "email": "updated_user@example.com",
        "given_name": "Jenny",
        "family_name": "Osborne",
        "metadata": {
            "salesforce_id": "EFGH56"
        }
    }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
    "customers": {
        "id": "CU123",
        "created_at": "2014-05-08T17:01:06.000Z",
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "email": "updated_user@example.com",
    "given_name": "Frank",
    "family_name": "Osborne",
    "address_line1": "29 Acer Road",
    "address_line2": "Apt 3",
    "address_line3": "Block 4",
    "city": "London",
    "region": null,
    "postal_code": "E8 3GX",
    "country_code": "GB",
    "language": "en",
    "phone_number": null,
    "swedish_identity_number": null,
    "danish_identity_number": null,
    "metadata": {
      "salesforce_id": "EFGH5678"
    }
  }
}

```

PUT https://api.gocardless.com/customers/CU123 HTTP/1.1

```

HTTP/1.1 410 Gone
Content-Type: application/json
{
  "error": {
    "documentation_url": "https://developer.gocardless.com/api-reference#customer_data_removed",
    "message": "This customer data has been removed",
    "type": "invalid_api_usage",
    "errors": [
      {
        "reason": "customer_removed",
        "message": "This customer data has been removed"
      }
    ],
    "code": 410,
    "request_id": "deadbeef-0000-4000-0000-444400004444"
  }
}

```

PHP

```

$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

```

```

$client->customers()->update("CU123", [
  'params' => ["email" => "tim@gocardless.com"]
]);

```

Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.customers.update("CU123", params={
  "email": "updated_user@example.com"
})

```

Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.customers.update(
  "CU123",
  params: {
    metadata: { custom_reference: "EXAMPLELTD01" }
  }
)

```

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.customers().update("CU123")
  .withEmail("updated_user@example.com")
  .withGivenName("Jenny")
  .execute();

```

JavaScript

```

const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const customer = await client.
  "CU123",
  {
    email: "updated_user@example.com"
  }
);

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.create(accessToken, Environment.SANDBOX);

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var customerRequest = new GoCardless.Services.CustomerUpdateRequest()
{
    Metadata = new Dictionary<string, string>()
    {
        {"custom_reference", "NEWREFERENCE001"}
    }
};
var customerResponse = await gocardless.Customers.UpdateAsync("CU0123", customerRequest);
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    customerUpdateParams := gocardless.CustomerUpdateParams{
        Email: "updated_user@example.com",
    }

    customer, err := client.Customers.Update(context, "CU123", customerUpdateParams)
}

```

## Remove a customer

Removed customers will not appear in search results or lists of customers (in our API or exports), and it will not be possible to load an individually removed customer by ID.

**The action of removing a customer cannot be reversed, so please use with care.**

Relative endpoint: `DELETE /customers/CU123`

Parameters

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

`DELETE https://api.gocardless.com/customers/CU123 HTTP/1.1`

`HTTP/1.1 204 No Content`

`DELETE https://api.gocardless.com/customers/CU123 HTTP/1.1`

`HTTP/1.1 410 Gone`

`Content-Type: application/json`

```
{
    "error": {
        "documentation_url": "https://developer.gocardless.com/api-reference#customer_data_removed",
        "message": "This customer data has been removed",
        "type": "invalid_api_usage",
        "errors": [
            {
                "reason": "customer_removed",
                "message": "This customer data has been removed"
            }
        ],
        "code": 410,
        "request_id": "deadbeef-0000-4000-0000-444400004444"
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->customers()->remove("CU123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client()

client.customers.remove("CU123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
    access_token: "your_access_token",
    environment: :sandbox
)
@client.customers.remove("CU123")
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.customers().remove("CU123").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customer = await client.customers.remove("CU123");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

await gocardless.Customers.RemoveAsync("CU0123");
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    customer, err := client.Customers.Remove(context, "CU123")
}
```

## Customer Bank Accounts

Customer Bank Accounts hold the bank details of a [customer](#). They always belong to a [customer](#), and may be linked to several Direct Debit [mandates](#).

Note that customer bank accounts must be unique, and so you will encounter a `bank_account_exists` error if you try to create a duplicate bank account. You may wish to handle this by updating the existing record instead, the ID of which will be provided as `links[customer_bank_account]` in the error response.

*Note:* To ensure the customer's bank accounts are valid, verify them first using [bank\\_details\\_lookups](#), before proceeding with creating the accounts

## Properties

<code>id</code>	Unique identifier, beginning with “BA”.
<code>account_holder_name</code>	Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a <a href="#">customer bank account token</a> .
<code>account_numberEnding</code>	The last few digits of the account number. Currently 4 digits for NZD bank accounts and 2 digits for other currencies.
<code>account_type</code>	Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See <a href="#">local details</a> for more information.
<code>bank_account_token</code>	A token to uniquely refer to a set of bank account details. This feature is still in early access and is only available for certain organisations.
<code>bank_name</code>	Name of bank, taken from the bank details.
<code>country_code</code>	<a href="#">ISO 3166-1 alpha-2 code</a> . Defaults to the country code of the iban if supplied, otherwise is required.
<code>created_at</code>	Fixed <a href="#">timestamp</a> , recording when this resource was created.
<code>currency</code>	<a href="#">ISO 4217 currency code</a> . Can be omitted.
<code>enabled</code>	Boolean value showing whether the bank account is active.
<code>metadata</code>	Key-value store of custom metadata.
<code>links[customer]</code>	ID of the <a href="#">customer</a> that owns the bank account.

### Create a customer bank account

Creates a new customer bank account.

There are three different ways to s...

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- [Local details](#)
- IBAN
- [Customer Bank Account Tokens](#)

For more information on the different fields required in each country, see [local bank details](#).

Relative endpoint: POST /customer\_bank\_accounts

**Note:** this endpoint is a legacy API endpoint and does not support GoCardless' newest features. We recommend using the [Billing Requests API](#) instead.

**Restricted:** this endpoint is [restricted](#) to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with approved payment pages.

#### Parameters

**account\_holder\_name**  
Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a [customer bank account token](#).

**account\_number**  
Bank account number - see [local details](#) for more information. Alternatively you can provide an [iban](#).

**account\_type**  
Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.

**bank\_code**  
Bank code - see [local details](#) for more information. Alternatively you can provide an [iban](#).

**branch\_code**  
Branch code - see [local details](#) for more information. Alternatively you can provide an [iban](#).

**country\_code**  
[ISO 3166-1 alpha-2 code](#). Defaults to the country code of the [iban](#) if supplied, otherwise is required.

**currency**  
[ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

**iban**  
International Bank Account Number. Alternatively you can provide [local details](#). IBANs are not accepted for Swedish bank accounts denominated in SEK - you must supply [local details](#).

**metadata**  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**links[customer]**  
*required* ID of the [customer](#) that owns this bank account.

**links[customer\_bank\_account\_token]**  
ID of a [customer bank account token](#) to use in place of bank account parameters.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com/customer\_bank\_accounts HTTP/1.1

Content-Type: application/json

```
{
  "customer_bank_accounts": {
    "account_number": "55779911",
    "branch_code": "200000",
    "account_holder_name": "Frank Osborne",
    "country_code": "GB",
    "links": {
      "customer": "CU123"
    }
  }
}
```

HTTP/1.1 201 Created  
Location: /customer\_bank\_accounts/BA123

Content-Type: application/json

```
{
  "customer_bank_accounts": {
    "id": "BA123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "account_holder_name": "Frank Osborne",
    "account_numberEnding": "11",
    "country_code": "GB",
    "currency": "GBP",
    "bank_name": "BARCLAYS BANK PLC",
    "metadata": {},
    "enabled": true,
    "links": {
      "customer": "CU123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment::SANDBOX;
]);
$client->customerBankAccounts([
  "params" => ["account_number" => "55779911", "branch_code" => "200000", "account_holder_name" => "Frank Osborne", "country_code" => "GB", "links" => [{"customer" => "CU123"}]
]);
```

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.customer_bank_accounts.create(params={
    "account_number": "55779911",
    "branch_code": "200000",
    "account_holder_name": "Frank Osborne",
    "country_code": "GB",
    "links": {
        "customer": "CU123"
    }
})
Ruby
```

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.customer_bank_accounts.create(
  params: {
    account_holder_name: "Fran Cosborne",
    account_number: "55779911",
    branch_code: "200000",
    country_code: "GB",
    links: {
      customer: "CU123"
    }
  }
)
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

CustomerBankAccount customerBankAccount = client.customerBankAccounts().create()
    .withAccountNumber("55779911")
    .withBranchCode("200000")
    .withAccountHolderName("Frank Osborne")
    .withCountryCode("GB")
    .withLinksCustomer("CU123")
    .execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customerBankAccount = await client.customerBankAccounts.create({
  account_number: "55779911",
  branch_code: "200000",
  account_holder_name: "Frank Osborne",
  country_code: "GB",
  links: {
    customer: "CU123"
  }
});
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerBankAccountRequest = new GoCardless.Services.CustomerBankAccountCreateRequest()
{
    AccountHolderName = "Example Ltd",
    AccountNumber = "55779911",
    BranchCode = "200000",
    CountryCode = "GB",
    Links = new GoCardless.Services.CustomerBankAccountCreateRequest.CustomerBankAccountLinks()
    {
        Customer = "CU0123"
    }
};

var customerBankAccountResponse = await gocardless.CustomerBankAccounts.CreateAsync(customerBankAccountRequest);
GoCardless.Resources.CustomerBankAccount customerBankAccount = customerBankAccountResponse.CustomerBankAccount;
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless"
)

func main() {
    accessToken := "your_access_token"
    config, err := gocardless.NewConfig(accessToken)
    if err != nil {
        fmt.Printf("got err in init: %v\n", err)
        return
    }
    client, err := gocardless.NewClient(config)
    if err != nil {
        fmt.Println("error in init: %v", err)
        return
    }
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

customerBankAccountCreateParams := gocardless.CustomerBankAccountCreateParams{
    AccountNumber: "55779911",
    BranchCode: "200000",
    AccountHolderName: "Frank Osborne",
    CountryCode: "GB",
    Links: gocardless.CustomerBankAccountCreateParamsLinks{
        Customer: "CU123",
    },
}

customerBankAccount, err := client.CustomerBankAccounts.Create(context, customerBankAccountCreateParams)
}

```

## List customer bank accounts

Returns a [cursor-paginated](#) list of your bank accounts.

Relative endpoint: GET /customer\_bank\_accounts

Parameters

`after`  
Cursor pointing to the start of the desired set.

`before`  
Cursor pointing to the end of the desired set.

`created_at[gt]`  
Limit to records created after the specified date-time.

`created_at[gte]`  
Limit to records created on or after the specified date-time.

`created_at[lt]`  
Limit to records created before the specified date-time.

`created_at[lte]`  
Limit to records created on or before the specified date-time.

`customer`  
Unique identifier, beginning with “CU”.

`enabled`  
Get enabled or disabled customer bank accounts.

`limit`  
Number of records to return.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/customer\_bank\_accounts HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "meta": {
        "cursors": {
            "before": null,
            "after": null
        },
        "limit": 50
    },
    "customer_bank_accounts": [
        {
            "id": "BA123",
            "created_at": "2014-05-08T17:01:06.000Z",
            "account_holder_name": "Frank Osborne",
            "account_numberEnding": "11",
            "account_type": null,
            "country_code": "GB",
            "currency": "GBP",
            "bank_name": "BARCLAYS BANK PLC",
            "metadata": {},
            "enabled": true,
            "links": {
                "customer": "CU123"
            }
        }
    ]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->customerBankAccounts()
```

```
$client->customerBankAccounts([
    'params' => ["customer" => "CU123", "enabled" => true]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client()

client.customer_bank_accounts.
```

Ruby

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.customer_bank_accounts.list

@client.customer_bank_accounts.list(
  params: {
    customer: "CU123",
    enabled: true
  }
)
Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

for (CustomerBankAccount customerBankAccount : client.customerBankAccounts().all().withCustomer("CU123").withEnabled(true).execute()) {
  System.out.println(customerBankAccount.getAccountHolderName());
}

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customerBankAccount = await client.customerBankAccounts.list();

// List all 'enabled' customer bank accounts.
await client.customerBankAccounts.list({ enabled: true });

```

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerBankAccountRequest = new GoCardless.Services.CustomerBankAccountListRequest()
{
  Customer = "CU000123"
};

var customerBankAccountListResponse = gocardless.CustomerBankAccounts.All(customerBankAccountRequest);
foreach (GoCardless.Resources.CustomerBankAccount customerBankAccount in customerBankAccountListResponse)
{
  Console.WriteLine(customerBank.Id);
}

```

## Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  customerBankAccountListParams := gocardless.CustomerBankAccountListParams{
    Enabled: true,
  }

  customerBankAccountListResult, err := client.CustomerBankAccounts.List(context, customerBankAccountListParams)
  for _, customerBankAccount := range customerBankAccountListResult.CustomerBankAccounts {
    fmt.Println(customerBankAccount.Id)
  }
}

```

**Get a single customer bank account**

Retrieves the details of an existing customer bank account.

Relative endpoint: GET /customer/{customer}/bank-accounts/{id}

○ ○ ○ ○ ○ ○ ○ ○  
HTTP PHP Python Ruby Java Java  
HTTP

GET https://api.gocardless.com/v2/customer/CU123/bank-accounts/BA123

HTTP/1.1 200 OK  
Content-Type: application/json  
{  
 "customer\_bank\_accounts": {  
 "id": "BA123",  
 },  
}

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "created_at": "2014-05-08T17:01:06.000Z",
    "account_numberEnding": "11",
    "account_holder_name": "Frank Osborne",
    "account_type": null,
    "country_code": "GB",
    "currency": "GBP",
    "bank_name": "BARCLAYS BANK PLC",
    "metadata": {},
    "enabled": true,
    "links": {
      "customer": "CU123"
    }
}
}

```

GET https://api.gocardless.com/customer\_bank\_accounts/BA123 HTTP/1.1

HTTP/1.1 410 Gone  
Content-Type: application/json

```

{
  "error": {
    "documentation_url": "https://developer.gocardless.com/api-reference#customer_data_removed",
    "message": "This customer data has been removed",
    "type": "invalid_api_usage",
    "errors": [
      {
        "reason": "customer_removed",
        "message": "This customer data has been removed"
      }
    ],
    "code": 410,
    "request_id": "deadbeef-0000-4000-0000-444400004444"
  }
}

```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->customerBankAccounts()->get("BA123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.customer_bank_accounts.get("BA123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.customer_bank_accounts.get("BA123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

CustomerBankAccount customerBankAccount = client.customerBankAccounts.get("BA123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customerBankAccount = await client.customerBankAccounts.find("BA123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerBankAccountResponse = await gocardless.CustomerBankAccounts.GetAsync("BA0123");
GoCardless.Resources.CustomerBankAccount customerBankAccount = customerBankAccountResponse.CustomerBankAccount;
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in initialising client. as , err.Error()")
    return
  }
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
}
```

```
customerBankAccount, err := client.CustomerBankAccounts.Get(context, "BA123")
```

```
}
```

## **Update a customer bank account**

Updates a customer bank account object. Only the metadata parameter is allowed.

Relative endpoint: PUT /customer\_bank\_accounts/BA123

### Parameters

## metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
PUT https://api.gocardless.com/customer_bank_accounts/BA123 HTTP/1.1
Content-Type: application/json
{
  "customer_bank_accounts": {
    "metadata": {
      "key": "value"
    }
  }
}
```

HTTP/1.1 200 OK  
Content-Type: application/json

```
{ "customer_bank_accounts": {  
    "id": "BA123",  
    "created_at": "2014-05-08T17:01:06.000Z",  
    "account_numberEnding": "11",  
    "account_holder_name": "Frank Osborne",  
    "account_type": null,  
    "country_code": "GB",  
    "currency": "GBP",  
    "bank_name": "BARCLAYS BANK PLC",  
    "metadata": {  
        "key": "value"  
    },  
    "enabled": true,  
    "links": {  
        "customer": "CU123"  
    }  
}
```

PUT https://api.gocardless.com/customer\_bank\_accounts/BA123 HTTP/1.1

HTTP/1.1 410 Gone  
Content-Type: application/json

```
{  
  "error": {  
    "documentation_url": "https://developer.gocardless.com/api-reference#customer_data_removed",  
    "message": "This customer data has been removed",  
    "type": "invalid_api_usage",  
    "errors": [  
      {  
        "reason": "customer_removed",  
        "message": "This customer data has been removed"  
      }  
    ],  
    "code": 410,  
    "request_id": "deadbeef-0000-4000-0000-444400004444"  
  }  
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->customerBankAccounts()->update("BA123", [
    "params" => ["metadata" => ["key" => "value"]]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client

client.customer_bank_accounts.  
    "metadata": {"key": "value"}  
})
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

params: {
  metadata: { description: "Business account" }
}
)
Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.customerBankAccounts().update("BA123")
  .withMetadata("description", "Business account")
  .execute();

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customerBankAccount = await client.customerBankAccounts.update(
  "BA123",
  {
    metadata: {
      key: "value"
    }
  }
);

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerBankAccountRequest = new GoCardless.Services.CustomerBankAccountUpdateRequest()
{
  Metadata = new Dictionary<string, string>()
  {
    {"description", "Business account"}
  }
};

var customerBankAccountResponse = await gocardless.CustomerBankAccounts.UpdateAsync("BA0123", customerBankAccountRequest);

Go

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  customerBankAccountUpdateParams := gocardless.CustomerBankAccountUpdateParams{
    Metadata: map[string]interface{}{"key": "value"},
  }

  customerBankAccount, err := client.CustomerBankAccounts.Update(context, customerBankAccountUpdateParams)
}

}

```

## Disable a customer bank account

Immediately cancels all associated mandates and cancellable payments.

This will return a `disable_failed` error if the bank account has already been disabled.

A disabled bank account can be re-enabled by creating a new bank account resource with the same details.

Relative endpoint: POST /customer\_bank\_accounts/BA123/actions/disable

HTTP PHP Python Ruby Java Java  
HTTP

POST https://api.gocardless.co

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "customer_bank_accounts": {
    "id": "BA123",
    "created_at": "2014-05-08T",
    "account_numberEnding": ".",
    "account_holder_name": "Fr",
    "account_type": null,
    "country_code": "GB",
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "currency": "GBP",
    "bank_name": "BARCLAYS BANK PLC",
    "metadata": {},
    "enabled": false,
    "links": {
      "customer": "CU123"
    }
  }
}
PHP

```

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

```

```
$client->customerBankAccounts()->disable("BA123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
client.customer_bank_accounts.disable("BA123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.customer_bank_accounts.disable("BA123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.customerBankAccounts.disable("BA123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);
```

```
const customerBankAccount = await client.customerBankAccounts.disable("BA123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerBankAccountResponse = await gocardless.CustomerBankAccounts.DisableAsync("BA0123");
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  customerBankAccount, err := client.CustomerBankAccounts.Disable(context, "BA123")
}
```

## Customer Notifications

Customer Notifications represent notifications are all identified in the list.

Note that these are ephemeral records.

**Restricted:** This API is currently

Properties

**id**

The id of the notification.

**action\_taken**

The action that was taken on the notification.

**action\_taken\_at**

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

esource and the customer to be

ion themselves.

Fixed [timestamp](#), recording when this action was taken.

**action\_taken\_by**  
A string identifying the integrator who was able to handle this notification.

**type**  
The type of notification the customer shall receive. One of:

- payment\_created
- payment\_cancelled
- mandate\_created
- mandate\_blocked
- subscription\_created
- subscription\_cancelled
- instalment\_schedule\_created
- instalment\_schedule\_cancelled

**links[customer]**  
The customer who should be contacted with this notification.

**links[event]**  
The event that triggered the notification to be scheduled.

**links[mandate]**  
The identifier of the related mandate.

**links[payment]**  
The identifier of the related payment.

**links[refund]**  
The identifier of the related refund.

**links[subscription]**  
The identifier of the related subscription.

## Handle a notification

“Handling” a notification means that you have sent the notification yourself (and don’t want GoCardless to send it). If the notification has already been actioned, or the deadline to notify has passed, this endpoint will return an `already_actioned` error and you should not take further action. This endpoint takes no additional parameters.

Relative endpoint: POST `/customer_notifications/PCN123/actions/handle`

Parameters

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/customer_notifications/PCN123/actions/handle HTTP/1.1
```

```
HTTP/1.1 201 Created
Content-Type: application/json
{
  "customer_notifications": {
    "id": "PCN123",
    "type": "payment_created",
    "action_taken": "handled",
    "action_taken_at": "2018-08-01T13:42:56.000Z",
    "action_taken_by": "Partner name",
    "links": {
      "customer": "CU123",
      "event": "EV123",
      "payment": "PM123"
    }
  }
}
```

```
POST https://api.gocardless.com/customer_notifications/PCN123/actions/handle HTTP/1.1
```

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
{
  "error": {
    "message": "You are not authorised to handle notifications of this type.",
    "documentation_url": "https://developer.gocardless.com/api-reference#notification_type_forbidden",
    "type": "gocardless",
    "code": 403,
    "request_id": "900ceabb-c5",
    "errors": [
      {
        "reason": "notification_type_forbidden",
        "message": "You are not authorised to handle notifications of this type."
      }
    ]
  }
}
```

```
POST https://api.gocardless.com/api-reference/
```

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"message": "You are not the notification owner for this resource.",
"documentation_url": "https://developer.gocardless.com/api-reference#not_notification_owner",
"type": "gocardless",
"code": 403,
"request_id": "900ceabb-c55d-4006-80b7-a7a40c171e31",
"errors": [
  {
    "reason": "not_notification_owner",
    "message": "You are not the notification owner for this resource."
  }
]
}
}

```

POST https://api.gocardless.com/customer\_notifications/PCN123/actions/handle HTTP/1.1

HTTP/1.1 410 Gone  
Content-Type: application/json

```
{
  "error": {
    "message": "This notification can no longer be handled.",
    "documentation_url": "https://developer.gocardless.com/api-reference#already_actioned",
    "type": "gocardless",
    "code": 410,
    "request_id": "900ceabb-c55d-4006-80b7-a7a40c171e31",
    "errors": [
      {
        "reason": "already_actioned",
        "message": "This notification can no longer be handled."
      }
    ]
  }
}
```

## PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
$client->customerNotifications()->handle("PCN123")
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.customer_notifications.handle("PCN123")
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.customer_notifications.handle("PCN123")
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

CustomerNotification customerNotification = client.customerNotifications().handle("PCN123").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const customerNotification = await client.customerNotifications.handle("PCN123");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customerNotificationResponse = await gocardless.CustomerNotifications.HandleAsync("PCN123");
GoCardless.Resources.CustomerNotification customerNotification = customerNotificationResponse.CustomerNotification;
```

## Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in initialising client: %s", err.Error())
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        return
    }

    customerNotificationHandleParams := map[string]interface{}{}
    customerNotification, err := client.CustomerNotifications.Handle(context, "PCN123", customerNotificationHandleParams)
}

}

```

## Events

Events are stored for all webhooks. An event refers to a resource which has been updated, for example a payment which has been collected, or a mandate which has been transferred. Event creation is an asynchronous process, so it can take some time between an action occurring and its corresponding event getting included in API responses. See [here](#) for a complete list of event types.

### Properties

**id**  
Unique identifier, beginning with “EV”.

**action**  
What has happened to the resource. See [Event Actions](#) for the possible actions.

**created\_at**  
Fixed [timestamp](#), recording when this resource was created.

**customer\_notifications**  
Present only in webhooks when an integrator is authorised to send their own notifications. See [here](#) for further information.

Each instance will contain these properties:

- **deadline**: Time after which GoCardless will send the notification by email.
- **id**: The id of the notification.
- **mandatory**: Whether or not the notification must be sent.
- **type**: See [here](#) for a complete list of customer notification types.

**details[bank\_account\_id]**  
When we send a creditor `new_payout_currency_added` webhook, we also send the bank account id of the new account

**details[cause]**  
What triggered the event. *Note*: cause is our simplified and predictable key indicating what triggered the event.

**details[currency]**  
When we send a creditor `new_payout_currency_added` webhook, we also send the currency of the new account

**details[description]**  
Human readable description of the cause. *Note*: Changes to event descriptions are not considered breaking.

**details[item\_count]**  
Count of rows in the csv. This is sent for export events

**details[not\_retried\_reason]**  
When `will_attempt_retry` is set to false, this field will contain the reason the payment was not retried. This can be one of:

- **failure\_filter\_applied**: The payment won’t be intelligently retried as there is a high likelihood of failure on retry.
- **other**: The payment won’t be intelligently retried due to any other reason.

**details[origin]**  
Who initiated the event. One of:

- **bank**: this event was triggered by a report from the banks
- **gocardless**: this event was performed by GoCardless automatically
- **api**: this event was triggered by an API endpoint
- **customer**: this event was triggered by a Customer
- **payer**: this event was triggered by a Payer

**details[property]**  
When we send a creditor `creditor_updated` webhook, this tells you which property on the creditor has been updated

**details[reason\_code]**  
Set when a bank is the origin of the event. This is the reason code received in the report from the customer’s bank. See the [GoCardless Direct Debit guide](#) for information on the meanings of different reason codes. *Note*: `reason_code` is payment scheme-specific and can be inconsistent between banks.

**details[scheme]**  
A bank payment scheme. Set when a bank is the origin of the event.

**details[will\_attempt\_retry]**  
Whether the payment will be retried automatically. Set on a payment failed event.

**metadata**  
The metadata that was passed

This field will only be populated

**resource\_metadata**  
The metadata of the resource received from performing a GET on the resource.

The resource type for this endpoint

- **billing\_requests**
- **creditors**
- **exports**

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

eld on the response you would

- instalment\_schedules
- mandates
- payer\_authorisations
- payments
- payouts
- refunds
- scheme\_identifiers
- subscriptions
- outbound\_payments

**links[bank\_authorisation]**  
ID of a [bank authorisation](#).

**links[billing\_request]**  
ID of a [billing request](#).

**links[billing\_request\_flow]**  
ID of a [billing request flow](#).

**links[creditor]**  
If resource\_type is creditor, this is the ID of the [creditor](#) which has been updated.

**links[customer]**  
ID of a [customer](#).

**links[customer\_bank\_account]**  
ID of a [customer bank account](#).

**links[instalment\_schedule]**  
If resource\_type is instalment\_schedule, this is the ID of the [instalment schedule](#) which has been updated.

**links[mandate]**  
If resource\_type is mandates, this is the ID of the [mandate](#) which has been updated.

**links[mandate\_request\_mandate]**  
If resource\_type is billing\_requests, this is the ID of the [mandate](#) which has been created.

**links[new\_customer\_bank\_account]**  
This is only included for mandate transfer events, when it is the ID of the [customer bank account](#) which the mandate is being transferred to.

**links[new\_mandate]**  
This is only included for mandate replaced events, when it is the ID of the new [mandate](#) that replaces the existing mandate.

**links[organisation]**  
If the event is included in a [webhook](#) to an [OAuth app](#), this is the ID of the account to which it belongs.

**links[parent\_event]**  
If this event was caused by another, this is the ID of the cause. For example, if a mandate is cancelled it automatically cancels all pending payments associated with it; in this case, the payment cancellation events would have the ID of the mandate cancellation event in this field.

**links[payer\_authorisation]**  
ID of a [payer authorisation](#).

**links[payment]**  
If resource\_type is payments, this is the ID of the [payment](#) which has been updated.

**links[payment\_request\_payment]**  
If resource\_type is billing\_requests, this is the ID of the [payment](#) which has been created for Instant Bank Payment.

**links[payout]**  
If resource\_type is payouts, this is the ID of the [payout](#) which has been updated.

**links[previous\_customer\_bank\_account]**  
This is only included for mandate transfer events, when it is the ID of the [customer bank account](#) which the mandate is being transferred from.

**links[refund]**  
If resource\_type is refunds, this is the ID of the [refund](#) which has been updated.

**links[scheme\_identifier]**  
If resource\_type is scheme\_identifiers, this is the ID of the [scheme identifier](#) which has been updated.

**links[subscription]**  
If resource\_type is subscription, this is the ID of the [subscription](#) which has been updated.

## List events

Returns a [cursor-paginated](#) list of your events.

Relative endpoint: GET /events

### Parameters

**action**  
Limit to events with a given action.

**after**  
Cursor pointing to the start of the desired set.

**before**  
Cursor pointing to the end of the desired set.

**billing\_request**  
ID of a [billing request](#). If specified:

**created\_at[gt]**  
Limit to records created after.

**created\_at[gte]**  
Limit to records created on or before.

**created\_at[lt]**  
Limit to records created before.

**created\_at[lte]**  
Limit to records created on or before.

**creditor**  
ID of an [creditor](#). If specified:

**export**

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité



```

"events": [
  {
    "id": "EV123",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "mandates",
    "action": "cancelled",
    "details": {
      "origin": "bank",
      "cause": "bank_account_disabled",
      "description": "Customer's bank account closed",
      "scheme": "bacs",
      "reason_code": "ADDACS-B"
    },
    "metadata": {},
    "resource_metadata": { "order_id": "123" },
    "links": {
      "mandate": "MD123"
    }
  },
  {
    "id": "EV456",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "payments",
    "action": "cancelled",
    "details": {
      "origin": "bank",
      "cause": "mandate_cancelled",
      "description": "The mandate for this payment was cancelled at a bank branch.",
      "scheme": "bacs",
      "reason_code": "ADDACS-B"
    },
    "metadata": {},
    "resource_metadata": { "order_dispatch_date": "2014-05-22" },
    "links": {
      "payment": "PM123",
      "parent_event": "EV123"
    }
  }
]
}

```

GET https://api.gocardless.com/events?resource\_type=payments&include=payment HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "after": null,
      "before": null
    },
    "limit": 50
  },
  "events": [
    {
      "id": "EV456",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payments",
      "action": "cancelled",
      "details": {
        "origin": "bank",
        "cause": "mandate_cancelled",
        "description": "The mandate for this payment was cancelled at a bank branch.",
        "scheme": "bacs",
        "reason_code": "ADDACS-B"
      },
      "metadata": {},
      "resource_metadata": { "order_dispatch_date": "2014-05-22" },
      "links": {
        "payment": "PM123",
        "parent_event": "EV123"
      }
    }
  ],
  "linked": {
    "payments": [
      {
        "id": "PM123",
        "created_at": "2014-05-08T17:01:06.000Z",
        "charge_date": "2014-05-15",
        "amount": 100,
        "description": null,
        "currency": "GBP",
        "status": "paid_out",
        "reference": "WINEBOX001",
        "metadata": {
          "order_dispatch_date": "2014-05-22"
        },
        "amount_refunded": 0,
        "links": {
          "mandate": "MD123",
          "creditor": "CR123"
        }
      }
    ]
  }
}

```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
$environment' => \GoCardlessPro\Environment::TEST;
]);
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
$client->events()->list();

$client->events()->list([
    "params" => ["resource_type" => "payments"]
]);
Python
```

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.events.list().records

client.events.list(params={"resource_type": "payments"}).records
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.events.list

@client.events.list(params: { resource_type: "payments" })

@client.events.list.records.each { |event| puts event.inspect }
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

for (Event event : client.events().all().withResourceType("payments").execute()) {
    System.out.println(event.getAction());
}
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const events = await client.events.list();

// List all events of a given resource type.
const events = await client.events.list({ resource_type: "payments" });
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var eventRequest = new GoCardless.Services.EventListRequest()
{
    ResourceType = GoCardless.Services.EventListRequest.EventResourceType.Payments
};

var eventListResponse = gocardless.Events.All(eventRequest);
foreach (GoCardless.Resources.Event eventResource in eventListResponse)
{
    Console.WriteLine(eventResource.Action);
}
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    eventListParams := gocardless.ResourceType{ "payments", }
    eventListResult, err := client.List(eventListParams)
    for _, event := range eventListResult {
        fmt.Println(event.Action)
    }
}
```

## Get a single event

Retrieves the details of a single event.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité



## Reconciling Payouts with Events

When a payout is created not only does it have an event created for it, but all payments and refunds within that payout have events created for them too. If you know the payout's ID, you can fetch all of these events using two API requests.

The first step is to find the “paid” event for the Payout you’re interested in. You can do this by filtering events based on the ID of the payout and the action, which is “paid”:

```
GET https://api.gocardless.com/events?payout=PY123&action=paid
```

To find child events for a payout, filter for events with a `parent_event` equal to the Payout’s paid event:

```
GET https://api.gocardless.com/events?parent_event=EV123
```

This will return a paginated list of all child events for payments and refunds associated with the payout.

If you’d also like to return the associated payments or refunds, you can scope the request to a `resource_type` and include those resources:

```
GET https://api.gocardless.com/events?parent_event=EV123&resource_type=payments&include=payment
```

```
GET https://api.gocardless.com/events?parent_event=EV123&resource_type=refunds&include=refund
```



HTTP  
HTTP

```
GET https://api.gocardless.com/events?payout=PY123&action=paid HTTP/1.1
```

HTTP/1.1 200 (OK)

Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "after": null,
      "before": null
    },
    "limit": 50
  },
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payouts",
      "action": "paid",
      "details": {
        "origin": "gocardless",
        "cause": "payout_paid",
        "description": "Payout sent"
      },
      "metadata": {},
      "links": {
        "payout": "PO123"
      }
    }
  ]
}
```

```
GET https://api.gocardless.com/events?parent_event=EV123 HTTP/1.1
```

HTTP/1.1 200 (OK)

Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "after": null,
      "before": null
    },
    "limit": 50
  },
  "events": [
    {
      "id": "EV456",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payments",
      "action": "paid_out",
      "details": {
        "origin": "gocardless",
        "cause": "payment_paid_out",
        "description": "The payment has been paid out by GoCardless."
      },
      "metadata": {},
      "links": {
        "payment": "PM123"
      }
    }
  ]
}
```

## Exports

File-based exports of data

Properties

`id`

Unique identifier, beginning



Consultez notre politique de confidentialité

`created_at`

Fixed [timestamp](#), recording when this resource was created.

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

`currency`  
The currency of the export (if applicable)  
`download_url`  
Download url for the export file. Subject to expiry.  
`export_type`  
The type of the export

## Get a single export

Returns a single export.

Relative endpoint: GET /exports/EX123

**Restricted:** This endpoint is restricted to GoCardless Embed customers. Please [contact us](#) if you are interested in using this product.

○ ○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/exports/EX123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "exports": {
    "id": "EX123",
    "created_at": "2014-01-01T12:00:00.000Z",
    "export_type": "payout_transactions_reconciliation",
    "download_url": "https://downloads.gocardless.com/EX123/example.csv",
    "currency": "GBP"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->exports()->get('EX123');
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.exports.get("EX123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.exports.get("EX123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
Export export = client.exports().get("EX123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const exportData = await client.exports.find("EX123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var exportResponse = await gocardless.Exports.GetAsync("EX123");
GoCardless.Resources.Export export = exportResponse.Export;
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-go"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.New(config)
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
}

export, err := client.Exports.Get(context, "EX123")
}

```

## List exports

Returns a list of exports which are available for download.

Relative endpoint: GET /exports

**Restricted:** This endpoint is restricted to GoCardless Embed customers. Please [contact us](#) if you are interested in using this product.

Parameters

**after**  
Cursor pointing to the start of the desired set.  
**before**  
Cursor pointing to the end of the desired set.  
**limit**  
Number of records to return.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/exports HTTP/1.1

```

HTTP/1.1 200 OK
Content-Type: application/json
{
"exports": [
    {
        "id": "EX123",
        "created_at": "2014-01-01T12:00:00.000Z",
        "export_type": "payout_transactions_reconciliation",
        "currency": "GBP"
    },
    {
        "id": "EX456",
        "created_at": "2014-01-01T12:00:00.000Z",
        "export_type": "payout_transactions_reconciliation",
        "currency": "EUR"
    }
]
}
```

PHP

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

```

\$client->exports()->list();

Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.exports.list().records

```

Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

```

@client.exports.list

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

```

```

for (Export export : client.exports.list())
    System.out.println(export.get());
}

```

JavaScript

```

const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');
const exports = await client.exports.list();

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var exportListResponse = gocardless.Exports.All();
foreach (GoCardless.Resources.Export export in exportListResponse)
{
    Console.WriteLine(export.Id);
}
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    exportListParams := gocardless.ExportListParams{}
    exportListResult, err := client.Exports.List(context, exportListParams)
    for _, export := range exportListResult.Exports {
        fmt.Println(export.Id)
    }
}

```

## Instalment Schedules

Instalment schedules are objects which represent a collection of related payments, with the intention to collect the `total_amount` specified. The API supports both schedule-based creation (similar to subscriptions) as well as explicit selection of differing payment amounts and charge dates.

Unlike subscriptions, the payments are created immediately, so the instalment schedule cannot be modified once submitted and instead can only be cancelled (which will cancel any of the payments which have not yet been submitted).

Customers will receive a single notification about the complete schedule of collection.

### Properties

`id` Unique identifier, beginning with “IS”.

`created_at` Fixed [timestamp](#), recording when this resource was created.

`currency` [ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

`metadata` Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`name` Name of the instalment schedule, up to 100 chars. This name will also be copied to the payments of the instalment schedule if you use schedule-based creation.

`payment_errors` If the status is `creation_failed`, this property will be populated with validation failures from the individual payments, arranged by the index of the payment that failed.

`status` One of:

- `pending`: we’re waiting for GC to create the payments
- `active`: the payments have been created, and the schedule is active
- `creation_failed`: payment creation failed
- `completed`: we have passed the date of the final payment and all payments have been collected
- `cancelled`: the schedule has been cancelled
- `errored`: one or more payments have failed

### `total_amount`

The total amount of the instalment schedule, defined as the sum of all individual payments, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR). If the requested payment amounts do not sum up correctly, a validation error will be returned.

### `links[customer]`

ID of the associated [customer](#)

### `links[mandate]`

ID of the associated [mandate](#)

### `payments`

Array of IDs of the associated payments

### Create (with dates)

Creates a new instalment schedule. Otherwise, please check out the [schedule-based creation](#).

The `instalments` property is an array of [Instalment](#) objects.

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

dates you wish to charge.



Consultez notre politique de confidentialité

It can take quite a while to create the associated payments, so the API will return the status as pending initially. When processing has completed, a subsequent GET request for the instalment schedule will either have the status success and link to the created payments, or the status error and detailed information about the failures.

Relative endpoint: POST /instalment\_schedules

#### Parameters

##### currency

*required* ISO 4217 currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

##### instalments

*required* An explicit array of instalment payments, each specifying at least an amount and charge\_date. See [create \(with dates\)](#).

##### name

*required* Name of the instalment schedule, up to 100 chars. This name will also be copied to the payments of the instalment schedule if you use schedule-based creation.

##### total\_amount

*required* The total amount of the instalment schedule, defined as the sum of all individual payments, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR). If the requested payment amounts do not sum up correctly, a validation error will be returned.

##### app\_fee

The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

##### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

##### payment\_reference

An optional reference that will appear on your customer’s bank statement. The character limit for this reference is dependent on the scheme.

**ACH** - 10 characters

**Autogiro** - 11 characters

**Bacs** - 10 characters

**BECS** - 30 characters

**BECS NZ** - 12 characters

**Betalingservice** - 30 characters

**Faster Payments** - 18 characters

**PAD** - scheme doesn’t offer references

**PayTo** - 18 characters

**SEPA** - 140 characters

Note that this reference must be unique (for each merchant) for the BECS scheme as it is a scheme requirement. **Restricted:** You can only specify a payment reference for Bacs payments (that is, when collecting from the UK) if you’re on the [GoCardless Plus, Pro or Enterprise packages](#). **Restricted:** You can not specify a payment reference for Faster Payments.

##### retry\_if\_possible

On failure, automatically retry payments using [intelligent retries](#). Default is false. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

##### links[mandate]

*required* ID of the associated [mandate](#) which the instalment schedule will create payments against.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/instalment_schedules HTTP/1.1
Content-Type: application/json
```

```
{
  "instalment_schedules": {
    "name": "Bike Invoice 271",
    "currency": "GBP",
    "total_amount": "2500",
    "instalments": [
      {
        "amount": "1500",
        "charge_date": "2019-12-14"
      },
      {
        "amount": "1000",
        "charge_date": "2020-05-01"
      }
    ],
    "metadata": {},
    "links": {
      "mandate": "MD123"
    }
  }
}
```

HTTP/1.1 201 (Created)

Location: /instalment\_schedules/IS123

Content-Type: application/json

```
{
  "instalment_schedules": {
    "id": "IS123",
    "status": "pending",
    "total_amount": "2500",
    "metadata": {},
    "payment_errors": {},
    "links": {
      "mandate": "MD123",
      "customer": "CU123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment::SANDBOX;
]);
$instalment_schedule = $client->instalmentSchedules()->createWithDates([
]);
```

#### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"params" => [
  "name" => "ACME Invoice 103",
  "total_amount" => 10000, // 100 GBP in pence, collected from the customer
  "app_fee" => 10, // Your 10 pence fee, applied to each instalment,
                    // to be paid out to you
  "currency" => 'GBP',
  "instalments" => [
    [
      "charge_date" => '2019-08-20',
      "amount" => 3400
    ],
    [
      "charge_date" => '2019-09-03',
      "amount" => 3400
    ],
    [
      "charge_date" => '2019-09-17',
      "amount" => 3200
    ],
    [
      "links" => [
        "mandate" => 'MD0000XH9A3T4C'
      ],
      "metadata" => []
    ],
    "headers" => [
      'Idempotency-Key' => 'random_instalment_schedule_specific_string'
    ]
  ]);

```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
```

```
instalment_schedule = client.instalment_schedules.create_with_dates(
```

```
  params={
    "name": "ACME Invoice 103",
    "total_amount": 10000, # 100 GBP in pence, collected from the customer
    "app_fee": 10, # Your 10 pence fee, applied to each instalment,
                  # to be paid out to you
    "currency": 'GBP',
    "instalments": [
      {
        "charge_date": '2019-08-20',
        "amount": 3400
      },
      {
        "charge_date": '2019-09-03',
        "amount": 3400
      },
      {
        "charge_date": '2019-09-17',
        "amount": 3200
      },
      [
        "links": {
          "mandate": 'MD0000XH9A3T4C'
        },
        "metadata": {}
      },
      headers={
        'Idempotency-Key': 'random_instalment_schedule_specific_string'
      }
    ]
  }
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
instalment_schedule = @client.instalment_schedules.create_with_dates(
  params: {
    name: "ACME Invoice 103",
    total_amount: 10_000, # 100 GBP in pence, collected from the customer
    app_fee: 10, # Your 10 pence fee, applied to each instalment,
                 # to be paid out to you
    currency: 'GBP',
    instalments: [
      {
        charge_date: '2019-08-20',
        amount: 3_400
      },
      {
        charge_date: '2019-09-03',
        amount: 3_400
      },
      {
        charge_date: '2019-09-17',
        amount: 3_200
      },
    ],
    links: {
      mandate: 'MD0000XH9A3T4C'
    },
    metadata: {}
  },
  headers: {
    'Idempotency-Key': 'random_instalment_schedule_specific_string'
  }
)
```

Java

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

int[] amountsArray = {3400, 3400, 3200};
String[] datesArray = {"2019-08-20", "2019-09-03", "2019-09-17"};

InstalmentSchedule instalmentSchedule = client.instalmentSchedules().createWithDates()
    .withTotalAmount(10000) // 100 GBP in Pence, collected from the end customer.
    .withAppFee(10) // Your 10 pence fee, applied to each instalment, to be
                    // paid out to you
    .withCurrency("GBP")
    .withAmounts(amountsArray)
    .withDates(datesArray)
    .withLinksMandate("MD0000YTKZY4J")
    .withMetadata("invoiceId", "001")
    .withIdempotencyKey("random_instalment_schedule_specific_string")
    .execute();

```

JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const instalmentSchedule = await client.instalmentSchedules.createWithDates(
{
    name: "ACME Invoice 103",
    total_amount: 10000, // 100 GBP in pence, collected from the customer
    app_fee: 10, // Your 10 pence fee, applied to each instalment,
                 // to be paid out to you
    currency: 'GBP',
    instalments: [
        {
            charge_date: '2019-08-20',
            amount: 3400
        },
        {
            charge_date: '2019-09-03',
            amount: 3400
        },
        {
            charge_date: '2019-09-17',
            amount: 3200
        }
    ],
    links: {
        mandate: 'MD0000XH9A3T4C'
    },
    metadata: {}
},
    "instalment_schedule_idempotency_key"
);

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

```

```

var createResponse = await gocardless.InstalmentSchedules.CreateWithDatesAsync(
    new InstalmentScheduleCreateWithDatesRequest
    {
        TotalAmount = 10000, // 100 GBP in pence, collected from the customer
        AppFee = 10, // Your 10 pence fee, applied to each instalment,
                     // to be paid out to you
        Currency = GoCardless.Services.InstalmentScheduleCreateWithDatesRequest.InstalmentScheduleCurrency.GBP,
        Name = "ACME Invoice 103",
        Instalments = new InstalmentScheduleInstalments[]
        {
            new InstalmentScheduleInstalments
            {
                ChargeDate = "2019-08-20",
                Amount = 3400
            },
            new InstalmentScheduleInstalments
            {
                ChargeDate = "2019-09-03",
                Amount = 3400
            },
            new InstalmentScheduleInstalments
            {
                ChargeDate = "2019-09-17",
                Amount = 3200
            },
        },
        Links = new GoCardless
            .Services
            .InstalmentSchedules
            .InstalmentScheduleCreateWithDatesRequest
            {
                Mandate = "MD0000YTKZY4J"
            },
        Metadata = new Dictionary<string, string>
        {
            {"Invoice ID", "001"}
        },
        IdempotencyKey = "random_instalment_schedule_specific_string"
    }
);

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    instalmentScheduleCreateWithDatesParams := gocardless.InstalmentScheduleCreateWithDatesParams{
        TotalAmount: 10000,
        AppFee: 10,
        Currency: "GBP",
        Instalments: []gocardless.InstalmentScheduleCreateWithDatesParamsInstalments{
            {
                Amount: 3400,
                ChargeDate: "2019-08-20",
            },
            {
                Amount: 3400,
                ChargeDate: "2019-09-03",
            },
            {
                Amount: 3400,
                ChargeDate: "2019-09-17",
            },
        },
        Links: gocardless.InstalmentScheduleCreateWithDatesParamsLinks{
            Mandate: "MD123",
        },
        Metadata: map[string]interface{}{"invoiceId": "001"},
    }

    requestOption := gocardless.WithIdempotencyKey("random_instalment_schedule_specific_string")
    instalmentSchedule, err := client.InstalmentSchedules().CreateWithDates(context, instalmentScheduleCreateWithDatesParams, requestOption)
}

```

## Create (with schedule)

Creates a new instalment schedule object, along with the associated payments. This API is recommended if you wish to use the GoCardless scheduling logic. For finer control over the individual dates, please check out the [alternative version](#).

It can take quite a while to create the associated payments, so the API will return the status as `pending` initially. When processing has completed, a subsequent GET request for the instalment schedule will either have the status `success` and link to the created payments, or the status `error` and detailed information about the failures.

Relative endpoint: POST /instalment\_schedules

Parameters

`currency`  
`required` ISO 4217 currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

`instalments[amounts]`  
`required` List of amounts of each instalment, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

`instalments[interval]`  
`required` Number of interval\_units between charge dates. Must be greater than or equal to 1.

`instalments[interval_unit]`  
`required` The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.

`instalments[start_date]`  
The date on which the first payment should be charged. Must be on or after the `mandate`’s `next_possible_charge_date`. When left blank and `month` or `day_of_month` are provided, this will be set to the date of the first payment. If created without `month` or `day_of_month` this will be set as the mandate’s `next_possible_charge_date`

`name`  
`required` Name of the instalment schedule, up to 100 chars. This name will also be copied to the payments of the instalment schedule if you use schedule-based creation.

`total_amount`  
`required` The total amount of the instalment schedule, defined as the sum of all individual payments, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR). If the requested payment amounts do not sum up correctly, a validation error will be returned.

`app_fee`  
The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence)

`metadata`  
Key-value store of custom data.

`payment_reference`  
An optional reference that will be included in payment notifications.

ACH - 10 characters

Autogiro - 11 characters

Bacs - 10 characters

BECS - 30 characters

BECS NZ - 12 characters

Betalingservice - 30 characters

Faster Payments - 18 characters

PAD - scheme doesn’t offer this

PayTo - 18 characters

SEPA - 140 characters

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

rs.

on the scheme.

Note that this reference must be unique (for each merchant) for the BECS scheme as it is a scheme requirement. **Restricted:** You can only specify a payment reference for Bacs payments (that is, when collecting from the UK) if you're on the [GoCardless Plus, Pro or Enterprise packages](#). **Restricted:** You can not specify a payment reference for Faster Payments.

**retry\_if\_possible**

On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

**links[mandate]**

*required* ID of the associated [mandate](#) which the instalment schedule will create payments against.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/installment_schedules HTTP/1.1
```

Content-Type: application/json

{

```
  "instalment_schedules": {
    "name": "Bike Invoice 271",
    "currency": "GBP",
    "total_amount": "2500",
    "instalments": {
      "start_date": "2019-12-14",
      "interval_unit": "monthly",
      "interval": 1,
      "amounts": [
        "1500",
        "1000"
      ]
    },
    "metadata": {},
    "links": {
      "mandate": "MD123"
    }
}
```

}

HTTP/1.1 201 Created

Location: /installment\_schedules/IS123

Content-Type: application/json

{

```
  "instalment_schedules": {
    "id": "IS123",
    "status": "pending",
    "total_amount": "2500",
    "metadata": {},
    "payment_errors": {},
    "links": {
      "mandate": "MD123",
      "customer": "CU123"
    }
}
```

}

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$instalment_schedule = $client->installmentSchedules()->createWithSchedule([
  "params" => [
    "name" => "ACME Invoice 103",
    "total_amount" => 10000, // 100 GBP in pence, collected from the customer
    "app_fee" => 10, // Your 10 pence fee, applied to each instalment,
                      // to be paid out to you
    "currency" => "GBP",
    "instalments" => [
      "start_date" => "2019-08-20",
      "interval_unit" => "weekly",
      "interval" => 2,
      "amounts" => [
        3400,
        3400,
        3200
      ],
      ],
    "links" => [
      "mandate" => "MD0000XH9A3T4C"
    ],
    "metadata" => []
  ],
  "headers" => [
    "Idempotency-Key" => "random_instalment_schedule_specific_string"
  ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client

instalment_schedule = client.installment_schedules.create(
  params={
    "name": "ACME Invoice 103",
    "total_amount": 10000,
    "app_fee": 10, # Your 10 pence fee, applied to each instalment,
                   # to be paid out to you
    "currency": "GBP",
    "instalments": [
      {"start_date": "2019-08-20",
       "interval_unit": "weekly",
       "interval": 2}
    ]
  }
)
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

```

        "amounts": [
            3400,
            3400,
            3200
        ]
    },
    "links": {
        "mandate": "MD0000XH9A3T4C"
    },
    "metadata": {}
}, headers={
    "Idempotency-Key": "random_instalment_schedule_specific_string"
}
)
Ruby

```

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

instalment_schedule = @client.instalment_schedules.create_with_schedule(
  params: {
    name: 'ACME Invoice 103',
    total_amount: 10_000, # 100 GBP in pence, collected from the customer
    app_fee: 10, # Your 10 pence fee, applied to each instalment,
                 # to be paid out to you
    currency: 'GBP',
    instalments: {
      start_date: '2019-08-20',
      interval_unit: 'weekly',
      interval: 2,
      amounts: [
        3_400,
        3_400,
        3_200
      ]
    }
  },
  links: {
    mandate: 'MD0000XH9A3T4C'
  },
  metadata: {}
),
headers: {
  'Idempotency-Key': 'random_instalment_schedule_specific_string'
}
)

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

int[] amountsArray = {3400, 3400, 3200};
InstalmentSchedule instalmentSchedule = client.instalmentSchedules().createWithSchedule()
  .withTotalAmount(10000) // 100 GBP in Pence, collected from the customer.
  .withAppFee(10) // Your 10 pence fee, applied to each instalment, to be
                  // paid out to you
  .withCurrency("GBP")
  .withIntervalUnit(IntervalUnit.WEEKLY)
  .withInterval(2)
  .withAmounts(amountsArray)
  .withLinksMandate("MD0000YTKZY4J")
  .withMetadata("invoiceId", "001")
  .withIdempotencyKey("random_instalment_schedule_specific_string")
  .execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const instalmentSchedule = await client.instalmentSchedules.createWithSchedule(
{
  name: "ACME Invoice 103",
  total_amount: 10000, // 100 GBP in pence, collected from the customer
  app_fee: 10, // Your 10 pence fee, applied to each instalment,
               // to be paid out to you
  currency: "GBP",
  instalments: {
    start_date: "2019-08-20",
    interval_unit: "weekly",
    interval: 2,
    amounts: [
      3400,
      3400,
      3200
    ]
  }
  links: {
    mandate: "MD0000XH9A3T4C"
  },
  metadata: {}
},
  "instalment_schedule_idempot"
);

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

int[][] amountsArray = new int[][]{ 3400, 3400, 3200 };
var createResponse = await gocardless.InstalmentSchedules.CreateWithScheduleAsync(
    new InstalmentScheduleCreateWithScheduleRequest
    {
        TotalAmount = 10000, // 100 GBP in pence, collected from the customer
        AppFee = 10, // Your 10 pence fee, applied to each instalment,
                    // to be paid out to you
        Currency = GoCardless.Services.InstalmentScheduleCreateWithScheduleRequest.InstalmentScheduleCurrency.GBP,
        Name = "ACME Invoice 103",
        Instalments = new GoCardless
            .Services
            .InstalmentScheduleCreateWithScheduleRequest
            .InstalmentScheduleInstalments
        {
            StartDate = "2019-08-20",
            IntervalUnit = InstalmentScheduleCreateWithScheduleRequest.InstalmentScheduleInstalments.InstalmentScheduleIntervalUnit.Weekly,
            Interval = 2,
            Amounts = amountsArray
        },
        Metadata = new Dictionary<string, string>
        {
            {"Invoice ID", "001"}
        },
        Links = new GoCardless
            .Services
            .InstalmentScheduleCreateWithScheduleRequest
            .InstalmentScheduleLinks
        {
            Mandate = "MD0000XH9A3T4C"
        },
        IdempotencyKey = "random_instalment_schedule_specific_string"
    }
);
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    instalmentScheduleCreateWithScheduleParams := gocardless.InstalmentScheduleCreateWithScheduleParams{
        TotalAmount: 10000,
        AppFee:      10,
        Currency:   "GBP",
        Instalments: gocardless.InstalmentScheduleCreateWithScheduleParamsInstalments{
            Interval:    2,
            IntervalUnit: "monthly",
            Amounts:     []int{3400, 3400, 3200},
        },
        Metadata: map[string]interface{}{"invoiceId": "001"},
    }

    requestOption := gocardless.WithIdempotencyKey("random_instalment_schedule_specific_string")
    instalmentSchedule, err := client.InstalmentSchedules().CreateWithSchedule(context, instalmentScheduleCreateWithScheduleParams, requestOption)
}

```

## List instalment schedules

Returns a [cursor-paginated](#) list of your instalment schedules.

Relative endpoint: GET /instalment\_schedules

### Parameters

`after` Cursor pointing to the start of the desired set.

`before` Cursor pointing to the end of the desired set.

`created_at[gt]` Limit to records created after.

`created_at[gte]` Limit to records created on or after.

`created_at[lt]` Limit to records created before.

`created_at[lte]` Limit to records created on or before.

`customer` ID of the associated [custom](#)er.

`limit` Number of records to return.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

ID of the associated [mandate](#) which the instalment schedule will create payments against.

#### status

At most five valid status values

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
GET https://api.gocardless.com/installment_schedules?after=IS123 HTTP/1.1
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "before": "IS000",
      "after": "IS456",
    },
    "limit": 50
  },
  "installment_schedules": [
    {
      "id": "IS123",
      "name": "Bike Invoice 271",
      "currency": "GBP",
      "status": "active",
      "total_amount": "2500",
      "metadata": {},
      "payment_errors": {},
      "links": {
        "mandate": "MD123",
        "payments": ["PM123", "PM345"],
        "customer": "CU123"
      }
    },
    {
      "id": "IS456",
      "name": "INV-7465",
      "currency": "GBP",
      "status": "cancelled",
      "total_amount": "3600",
      "metadata": {},
      "payment_errors": {},
      "links": {
        "mandate": "MD456",
        "payments": ["PM567", "PM789"],
        "customer": "CU456"
      }
    }
  ]
}
```

#### PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->installmentSchedules()->list();

$client->installmentSchedules()->list([
  'params' => ["created_at[gt]" => "2015-11-03T09:30:00Z"]
]);
```

#### Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.installment_schedules.list()

client.installment_schedules.list(params={"after": "IS123"}).records
```

#### Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.installment_schedules.list

@client.installment_schedules.list(
  params: {
    "created_at[gt]" => "2016-08-06T09:30:00Z"
  }
)
```

```
@client.installment_schedules.list records_each do |instalment_schedule|
  puts instalment_schedule.inspect
end
```

#### Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

for (InstalmentSchedule instalmentSchedule : client.list("IS123")) {
  System.out.println(instalmentSchedule);
}
```

#### JavaScript

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const instalmentSchedules = await client.instalmentSchedules.list();

// List all instalment schedules associated with a given customer.
const instalmentSchedules = await client.instalmentSchedules.list({ customer: "CU123" });

.NET
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var instalmentScheduleListResponse = gocardless.InstalmentSchedules.All();
foreach (GoCardless.Resources.InstalmentSchedule instalmentSchedule in instalmentScheduleListResponse)
{
    Console.WriteLine(instalmentSchedule.Name);
}

Go
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    instalmentScheduleListParams := gocardless.InstalmentScheduleListParams{}
    instalmentScheduleListResult, err := client.InstalmentSchedules.List(context, instalmentScheduleListParams)
    for _, instalmentSchedule := range instalmentScheduleListResult.InstalmentSchedules {
        fmt.Println(instalmentSchedule.Name)
    }
}

```

## Get a single instalment schedule

Retrieves the details of an existing instalment schedule.

Relative endpoint: GET /instalment\_schedules/IS123

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/instalment\_schedules/IS123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "instalment_schedules": {
        "id": "IS123",
        "name": "INV-4142",
        "currency": "GBP",
        "status": "active",
        "total_amount": "2500",
        "metadata": {},
        "payment_errors": {},
        "links": {
            "mandate": "MD456",
            "payments": ["PM123", "PM345"],
            "customer": "CU456"
        }
    }
}
```

```
{
    "instalment_schedules": {
        "id": "IS123",
        "name": "INV-4142",
        "currency": "GBP",
        "status": "creation_failed",
        "total_amount": "2500",
        "metadata": {},
        "payment_errors": {
            "0": { "field": "descriptor" },
            "1": {}
        }
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
'access_token' => 'your_access_token';
'environment' => \GoCardless\Environment::SANDBOX;
]);
$client->instalmentSchedules()->get("IS123");
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.instalment_schedules.get("IS123")
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.instalment_schedules.get("IS123")
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

InstalmentSchedule instalmentSchedule = client.instalmentSchedules().get("IS123").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const instalmentSchedule = await client.instalmentSchedules.find("IS123");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var instalmentScheduleResponse = await gocardless.InstalmentSchedules.GetAsync("IS123");
GoCardless.Resources.Customer instalmentSchedule = instalmentScheduleResponse.InstalmentSchedule;
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }
    instalmentSchedule, err := client.InstalmentSchedules.Get(context, "IS123")
}
```

**Update an instalment schedule**

Updates an instalment schedule. This accepts only the metadata parameter.

Relative endpoint: PUT /instalment\_schedules/IS123

## Parameters

## metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

PUT https://api.gocardless.com/instalment\_schedules/IS123 HTTP/1.1

Content-Type: application/json

```
{
  "instalment_schedules": {
    "metadata": {
      "key": "value"
    }
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "instalment_schedules": {
    "id": "IS123",
    "name": "INV-4142",
    "currency": "GBP",
    "status": "active",
    "total_amount": "2500",
  }
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "metadata": {
      "key": "value"
    },
    "payment_errors": {},
    "links": {
      "mandate": "MD456",
      "payments": ["PM123", "PM345"],
      "customer": "CU456"
    }
  }
}

PHP
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->instalmentSchedules()->update("IS123", [
  'params' => ["metadata" => ["key" => "value"]]
]);

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.instalment_schedules.update("IS123", params={
  "metadata": {"key": "value"}
})

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.instalment_schedules.update(
  "IS123",
  params: {
    metadata: { key: "value" }
  }
)

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.instalmentSchedules().update("IS123")
  .withMetadata("key", "value")
  .execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const instalmentSchedule = await client.instalmentSchedules.update(
  "IS123",
  {
    metadata: {
      key: "value"
    }
  }
);

```

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var instalmentScheduleRequest = new GoCardless.Services.InstalmentScheduleUpdateRequest()
{
  Metadata = new Dictionary<string, string>()
  {
    {"key", "value"}
  }
};

var instalmentScheduleResponse = await gocardless.InstalmentSchedules.UpdateAsync("IS123", instalmentScheduleRequest);

```

## Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Printf("got err in client: %v\n", err)
    return
  }
}

```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    fmt.Println("error in initialising client: %s", err.Error())
    return
}

instalmentScheduleUpdateParams := gocardless.InstalmentScheduleUpdateParams{
    Metadata: map[string]interface{}{"key": "value"}
}

instalmentSchedule, err := client.InstalmentSchedules.Update(context, "IS123", instalmentScheduleUpdateParams)
}

```

## Cancel an instalment schedule

Immediately cancels an instalment schedule; no further payments will be collected for it.

This will fail with a `cancellation_failed` error if the instalment schedule is already cancelled or has completed.

Relative endpoint: POST /instalment\_schedules/IS123/actions/cancel

Parameters



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST [https://api.gocardless.com/instalment\\_schedules/IS123/actions/cancel](https://api.gocardless.com/instalment_schedules/IS123/actions/cancel) HTTP/1.1

```

HTTP/1.1 200 OK
Content-Type: application/json
{
    "instalment_schedules": {
        "id": "IS123",
        "name": "Bike Invoice 271",
        "currency": "GBP",
        "status": "cancelled",
        "total_amount": "2500",
        "metadata": {},
        "payment_errors": {},
        "links": {
            "mandate": "MD123",
            "payments": ["PM123", "PM345"],
            "customer": "CU123"
        }
    }
}

```

PHP

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->instalmentSchedules()->cancel("IS123");

```

Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.instalment_schedules.cancel("IS123")

```

Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.instalment_schedules.cancel("IS123")

```

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.instalmentSchedules().cancel("IS123").execute();

```

JavaScript

```

const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

```

```

const instalmentScheduleResponse = await client
    .instalmentSchedules()
    .cancel("IS123");

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(Environment.SANDBOX)
    .build();

var response = await gocardless
    .instalmentSchedules()
    .cancel("IS123");

```

Go

```

package main

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    instalmentScheduleCancelParams = make(map[string]interface{})
    instalmentSchedule, err := client.InstalmentSchedules.Cancel(context, "IS123", instalmentScheduleCancelParams)
}

```

## Logos

Logos are image uploads that, when associated with a creditor, are shown on the [billing request flow](#) payment pages.

### Properties

**id** Unique identifier, beginning with “LO”.

### Create a logo associated with a creditor

Creates a new logo associated with a creditor. If a creditor already has a logo, this will update the existing logo linked to the creditor.

We support JPG and PNG formats. Your logo will be scaled to a maximum of 300px by 40px. For more guidance on how to upload logos that will look great across your customer payment page and notification emails see [here](#).

Relative endpoint: POST /branding/logos

**Restricted:** This endpoint is restricted to GoCardless Embed customers. Please [contact us](#) if you are interested in using this product.

### Parameters

**image** required Base64 encoded string.

**links[creditor]** ID of the creditor the logo belongs to

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```

POST https://api.gocardless.com/branding/logos HTTP/1.1
Content-Type: application/json
{
    "logos": [
        "image": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgAAAAACAYAAQDgdz34AAAABmJLR0QA/wD/AP+gvaeTAAAA",
        "links": {
            "creditor": "CR123"
        }
    ]
}

```

HTTP/1.1 201 OK

```

Content-Type: application/json
{
    "logos": [
        "id": "L0123"
    ]
}

```

### PHP

```

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

```

```

$client->logos()->createForCreditor([
    "params" => [
        "image" => "data:image/png",
        "links" => [
            "creditor" => "CR123"
        ]
    ]
]);

```

### Python

```

import gocardless_pro
client = gocardless_pro.Client(
    "image": "data:image/png;bas",
    "links": {
        "creditor": "CR123"
    }
);

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    }
})
Ruby

```

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.logos.create_for_creditor(
  params: {
    image: "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgAAAAYCAYAADgdz34AAAABmJLR0QA/wD/AP+gvaeTAAAA",
    links: {
      creditor: "CR123"
    }
  }
)

```

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

Logo logo = client.logos().createForCreditor()
  .withImage("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgAAAAYCAYAADgdz34AAAABmJLR0QA/wD/AP+gvaeTAAAA")
  .withLinksCreditor("CR123")
  .execute();

```

JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const logo = await client.logos.createForCreditor({
  image: "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgAAAAYCAYAADgdz34AAAABmJLR0QA/wD/AP+gvaeTAAAA",
  links: {
    creditor: "CR123"
  }
});

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var logoCreateForCreditorRequest = new GoCardless.Services.LogoCreateForCreditorRequest()
{
  Image = "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgAAAAYCAYAADgdz34AAAABmJLR0QA/wD/AP+gvaeTAAAA",
  Links = new GoCardless.Services.LogoCreateForCreditorRequest.LogoLinks()
  {
    Creditor = "CR0123"
  }
};

var logoCreateForCreditorResponse = await gocardless.Logos.CreateForCreditorAsync(logoCreateForCreditorRequest);
GoCardless.Resources.Logo creditorLogo = logoCreateForCreditorResponse.LogoResponse;

```

Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  logoCreateForCreditorParams := gocardless.LogoCreateForCreditorParams{
    Image: "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgAAAAYCAYAADgdz34AAAABmJLR0QA/wD/AP+gvaeTAAAA",
    Links: gocardless.LogoCreateForCreditorParamsLinks{
      Creditor: "CR123"
    }
  }
  logo, err := client.Logos.CreateForCreditor(logoCreateForCreditorParams)
}

```

## Mandates

Mandates represent the Direct Debit mandate.

GoCardless will notify you via a

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Properties

**id**

Unique identifier, beginning with “MD”. Note that this prefix may not apply to mandates created before 2016.

**authorisation\_source**

This field is ACH specific, sometimes referred to as [SEC code](#).

This is the way that the payer gives authorisation to the merchant. web: Authorisation is Internet Initiated or via Mobile Entry (maps to SEC code: WEB) telephone: Authorisation is provided orally over telephone (maps to SEC code: TEL) paper: Authorisation is provided in writing and signed, or similarly authenticated (maps to SEC code: PPD)

**consent\_parameters[end\_date]**

The latest date at which payments can be taken, must occur after start\_date if present

**consent\_parameters[max\_amount\_per\_payment]**

The maximum amount that can be charged for a single payment

**consent\_parameters[max\_amount\_per\_period]**

The maximum total amount that can be charged for all payments in this period

**consent\_parameters[max\_payments\_per\_period]**

The maximum number of payments that can be collected in this period

**consent\_parameters[period]**

The repeating period for this mandate

**consent\_parameters[start\_date]**

The date from which payments can be taken

**consent\_type**

(Optional) Specifies the type of authorisation agreed between the payer and merchant. It can be set to one-off, recurring or standing for ACH, or single, recurring and sporadic for PAD.

**created\_at**

Fixed [timestamp](#), recording when this resource was created.

**funds\_settlement**

This field will decide how GoCardless handles settlement of funds from the customer.

- managed will be moved through GoCardless’ account, batched, and payed out.

- direct will be a direct transfer from the payer’s account to the merchant where invoicing will be handled separately.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**next\_possible\_charge\_date**

The earliest date that can be used as a charge\_date on any newly created payment for this mandate. This value will change over time.

**next\_possible\_standard\_ach\_charge\_date**

If this is an an ACH mandate, the earliest date that can be used as a charge\_date on any newly created payment to be charged through standard ACH, rather than Faster ACH. This value will change over time.

It is only present in the API response for ACH mandates.

**payments\_require\_approval**

Boolean value showing whether payments and subscriptions under this mandate require approval via an automated email before being processed.

**reference**

Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

**scheme**

Bank payment scheme to which this mandate and associated payments are submitted. Can be supplied or automatically detected from the customer’s bank account.

**status**

One of:

- pending\_customer\_approval: the mandate has not yet been signed by the second customer
- pending\_submission: the mandate has not yet been submitted to the customer’s bank
- submitted: the mandate has been submitted to the customer’s bank but has not been processed yet
- active: the mandate has been successfully set up by the customer’s bank
- suspended\_by\_payer: the mandate has been suspended by payer
- failed: the mandate could not be created
- cancelled: the mandate has been cancelled
- expired: the mandate has expired due to dormancy
- consumed: the mandate has been consumed and cannot be reused (note that this only applies to schemes that are per-payment authorised)
- blocked: the mandate has been blocked and payments cannot be created

**verified\_at**

[Timestamp](#) recording when this mandate was verified.

**links[creditor]**

ID of the associated [creditor](#)

**links[customer]**

ID of the associated [customer](#)

**links[customer\_bank\_account]**

ID of the associated [customer\\_bank\\_account](#)

**links[new\_mandate]**

ID of the new mandate if th

**Create a mandate**

Creates a new mandate object.

Relative endpoint: POST /mandate

**Nous utilisons des cookies**

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**Note:** this endpoint is a legacy API endpoint and does not support GoCardless' newest features. We recommend using the [Billing Requests API](#) instead.

**Restricted:** this endpoint is [restricted](#) to [GoCardless Pro](#) and [GoCardless Enterprise](#) accounts with approved payment pages.

**Warning:** by default, the ability to provide a custom mandate reference is switched off. The banking system rules for valid references are quite complex, and we recommend allowing GoCardless to generate it. If you would like to provide custom references, please contact support.

#### Parameters

##### authorisation\_source

This field is ACH specific, sometimes referred to as [SEC code](#).

This is the way that the payer gives authorisation to the merchant. web: Authorisation is Internet Initiated or via Mobile Entry (maps to SEC code: WEB) telephone: Authorisation is provided orally over telephone (maps to SEC code: TEL) paper: Authorisation is provided in writing and signed, or similarly authenticated (maps to SEC code: PPD)

##### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

##### payer\_ip\_address

For ACH customers only. Required for ACH customers. A string containing the IP address of the payer to whom the mandate belongs (i.e. as a result of their completion of a mandate setup flow in their browser).

Not required for creating offline mandates where `authorisation_source` is set to telephone or paper.

##### reference

Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

##### scheme

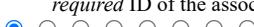
Bank payment scheme to which this mandate and associated payments are submitted. Can be supplied or automatically detected from the customer's bank account.

##### links[creditor]

ID of the associated [creditor](#). Only required if your account manages multiple creditors.

##### links[customer\_bank\_account]

*required* ID of the associated [customer bank account](#) which the mandate is created and submits payments against.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/mandates HTTP/1.1
Content-Type: application/json
{
  "mandates": {
    "scheme": "bacs",
    "metadata": {
      "contract": "ABCD1234"
    },
    "links": {
      "customer_bank_account": "BA123",
      "creditor": "CR123"
    }
  }
}
```

```
HTTP/1.1 201 Created
Location: /mandates/MD123
Content-Type: application/json
{
  "mandates": {
    "id": "MD123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "reference": "REF-123",
    "status": "pending_submission",
    "scheme": "bacs",
    "next_possible_charge_date": "2014-11-10",
    "metadata": {
      "contract": "ABCD1234"
    },
    "links": {
      "customer_bank_account": "BA123",
      "creditor": "CR123",
      "customer": "CU123"
    }
  }
}
```

#### PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->mandates()->create([
  'params' => [
    'scheme' => "ba",
    "metadata" => [
      "links" => [
        "cu"
      ]
    ]
  ]
]);
```

#### Python

```
import gocardless_pro
client = gocardless_pro.Client()
client.mandates.create(params=
  "scheme": "bacs",
  "metadata": [
    "contract": "ABCD1234"
  ],
},
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"links": {
  "customer_bank_account": "BA123",
  "creditor": "CR123"
}
})
Ruby

```

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

```

```

@client.mandates.create(
  params: {
    scheme: "bacs",
    links: {
      customer_bank_account: "BA123"
    }
  }
)

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

```

```

Mandate mandate = client.mandates().create()
  .withScheme("bacs")
  .withLinksCustomerBankAccount("BA123")
  .withMetadata("contract", "ABCD1234")
  .execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const mandate = await client.mandates.create({
  scheme: "bacs",
  metadata: {
    contract: "ABCD1234"
  },
  links: {
    customer_bank_account: "BA123",
    creditor: "CR123"
  }
});

```

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var mandateRequest = new GoCardless.Services.MandateCreateRequest()
{
  Links = new GoCardless.Services.MandateCreateRequest.MandateLinks()
  {
    CustomerBankAccount = "BA0123"
  },
  Metadata = new Dictionary<string, string>()
  {
    {"internal_reference", "ref_09011991"}
  },
  Scheme = "bacs"
};

var mandateResponse = await gocardless.Mandates.CreateAsync(mandateRequest);
GoCardless.Resources.Mandate mandate = mandateResponse.Mandate;

```

## Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s". err.Error())
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in init")
    return
  }

  mandateCreateParams := gocardless.MandateCreateParams{
    Scheme: "bacs",
    Links: gocardless.MandateCreateParams.MandateLinks{
      CustomerBankAccount: "BA123",
      Creditor: "CR123"
    },
    Metadata: map[string]interface{}{}
  }
}

```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
mandate, err := client.Mandates().Create(context, mandateCreateParams)
}
```

## List mandates

Returns a [cursor-paginated](#) list of your mandates.

Relative endpoint: GET /mandates

Parameters

```
after
    Cursor pointing to the start of the desired set.
before
    Cursor pointing to the end of the desired set.
created_at[gt]
    Limit to records created after the specified date-time.
created_at[gte]
    Limit to records created on or after the specified date-time.
created_at[lt]
    Limit to records created before the specified date-time.
created_at[lte]
    Limit to records created on or before the specified date-time.
creditor
    ID of a creditor. If specified, this endpoint will return all mandates for the given creditor. Cannot be used in conjunction with customer or customer_bank_account.
customer
    ID of a customer. If specified, this endpoint will return all mandates for the given customer. Cannot be used in conjunction with customer_bank_account or creditor.
customer_bank_account
    ID of a customer bank account. If specified, this endpoint will return all mandates for the given bank account. Cannot be used in conjunction with customer or creditor.
limit
    Number of records to return.
mandate_type
    Mandate type
reference
    Unique reference. Different schemes have different length and character set requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.
scheme
    Scheme you'd like to retrieve mandates for
status
    At most four valid status values
```



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/mandates HTTP/1.1

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "mandates": [
    {
      "id": "MD123",
      "created_at": "2014-05-08T17:01:06.000Z",
      "reference": "REF-123",
      "status": "pending_submission",
      "scheme": "bacs",
      "mandate_type": "bank_debit",
      "next_possible_charge_date": "2014-11-10",
      "metadata": {
        "contract": "ABCD1234"
      },
      "links": {
        "customer_bank_account": "BA123",
        "creditor": "CR123",
        "customer": "CU123"
      }
    }
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment::TEST;
]);
$client->mandates()->list();
$client->mandates()->list([
  'params' => ['customer' => 'CU123']
]);
```

Python

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandates.list().records

client.mandates.list(params={"customer": "CU123"}).records
Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.mandates.list

@client.mandates.list(params: { customer: "CU123" })

@client.mandates.list.records.each { |mandate| puts mandate.inspect }

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

for (Mandate mandate : client.mandates().all().withCustomer("CU123").execute()) {
    System.out.println(mandate.getId());
}

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const mandates = await client.mandates.list();

// List all mandates associated with a given customer.
const mandates = await client.mandates.list({ customer: "CU123" });

```

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var mandateRequest = new GoCardless.Services.MandateListRequest()
{
    Customer = "CU000123"
};

var mandateListResponse = gocardless.Mandates.All(mandateRequest);
foreach (GoCardless.Resources.Mandate mandate in mandateListResponse)
{
    Console.WriteLine(mandate.Id);
}

```

## Go

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    mandateListParams := gocardless.MandateListParams{
        Customer: "CU123",
    }

    mandateListResult, err := client.Mandates.List(context, mandateListParams)
    for _, mandate := range mandateListResult.Mandates {
        fmt.Println(mandate.Id)
    }
}

```

**Get a single mandate**

Retrieves the details of an existing mandate.

Relative endpoint: GET /mandates/{id}

HTTP PHP Python Ruby Java Java  
HTTP

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
GET https://api.gocardless.com/mandates/MD123 HTTP/1.1
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "mandate": {
    "id": "MD123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "reference": "REF-123",
    "status": "pending_submission",
    "scheme": "bacs",
    "mandate_type": "bank_debit",
    "next_possible_charge_date": "2014-11-10",
    "metadata": {
      "contract": "ABCD1234"
    },
    "links": {
      "customer_bank_account": "BA123",
      "creditor": "CR123",
      "customer": "CU123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->mandates()->get("MD123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
client.mandates.get("MD123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.mandates.get("MD123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
Mandate mandate = client.mandates().get("MD123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const mandate = await client.mandates.find("MD123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var mandateResponse = await gocardless.Mandates.GetAsync("MD0123");
GoCardless.Resources.Mandate mandate = mandateResponse.Mandate;
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in init\n")
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in init\n")
    return
  }
  mandate, err := client.Mandates.Get("MD123")
  if err != nil {
    fmt.Println("error in get\n")
    return
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

## Update a mandate



Consultez notre politique de confidentialité

Updates a mandate object. This accepts only the metadata parameter.

Relative endpoint: PUT /mandates/MD123

#### Parameters

##### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

PUT https://api.gocardless.com/mandates/MD123 HTTP/1.1

Content-Type: application/json

```
{
  "mandates": {
    "metadata": {
      "key": "value"
    }
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "mandates": {
    "id": "MD123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "reference": "REF-123",
    "status": "pending_submission",
    "scheme": "bacs",
    "next_possible_charge_date": "2014-11-10",
    "metadata": {
      "key": "value"
    },
    "links": {
      "customer_bank_account": "BA123",
      "creditor": "CR123",
      "customer": "CU123"
    }
  }
}
```

#### PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->mandates()->update("MD123", [
  "params" => ["metadata" => ["key" => "value"]]
]);
```

#### Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandates.update("MD123", params={
  "metadata": {"key": "value"}
})
```

#### Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.mandates.update(
  "MD123",
  params: {
    metadata: { contract_id: "ref_09011991" }
  }
)
```

#### Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.mandates().update("MD123").withMetadata("contract_id", "ref_09011991").execute();
```

#### JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const mandate = await client.mandates().update("MD123", {
  metadata: {"key": "value"}
});
```

#### .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var mandateRequest = new GoCardless.Services.MandateUpdateRequest()
{
    Metadata = new Dictionary<string, string>()
    {
        {"internal_reference", "ref_09011991"}
    }
};

var mandateResponse = await gocardless.Mandates.UpdateAsync("MD0123", mandateRequest);
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    mandateUpdateParams := gocardless.MandateUpdateParams{
        Metadata: map[string]interface{}{"key": "value"},
    }

    mandate, err := client.Mandates.Update(context, "MD123", mandateUpdateParams)
}

```

## Cancel a mandate

Immediately cancels a mandate and all associated cancellable payments. Any metadata supplied to this endpoint will be stored on the mandate cancellation event it causes.

This will fail with a `cancellation_failed` error if the mandate is already cancelled.

Relative endpoint: POST /mandates/MD123/actions/cancel

Parameters

`metadata`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/mandates/MD123/actions/cancel HTTP/1.1
Content-Type: application/json
{
    "data": {
        "metadata": {}
    }
}
```

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "mandates": {
        "id": "MD123",
        "created_at": "2014-05-08T17:01:06.000Z",
        "reference": "REF-123",
        "status": "cancelled",
        "scheme": "bacs",
        "next_possible_charge_date": null,
        "metadata": {
            "contract": "ABCD1234"
        },
        "links": {
            "customer_bank_account": "BA123",
            "creditor": "CR123",
            "customer": "CU123"
        }
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
['access_token' => 'your_access_token',
 'environment' => \GoCardlessPro\Environment::SANDBOX];
]);
```

```
$client->mandates()->cancel("MD123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client()
client.mandates.cancel("MD123")
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Ruby

```
@client = GoCardlessPro::Client.new(  
  access_token: "your_access_token",  
  environment: :sandbox  
)  
  
@client.mandates.cancel("MD123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();
```

```
client.mandates().cancel("MD123").execute()
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants Environments.Sandbox);

const mandateResponse = await client.mandates.cancel("MD123");
```

NET

```
String accessToken = "your_access_token";
GoCardlessClient goCardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);
```

```
var mandateResponse = await gocardless.Mandates.CancelAsync("MD0123");
```

Go

```
package main
```

```
import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    mandateCancelParams := gocardless.MandateCancelParams{}
    mandate, err := client.Mandates.Cancel(context, "MD123", mandateCancelParams)
}
```

#### **Reinstate a mandate**

Reinstates a cancelled or expired mandate to the banks. You will receive a `resubmission_requested` webhook, but after that reinstating the mandate follows the same process as its initial creation, so you will receive a `submitted` webhook, followed by a `reinstated` or `failed` webhook up to two working days later. Any metadata supplied to this endpoint will be stored on the `resubmission_requested` event it causes.

This will fail with a `mandate_not_inactive` error if the mandate is already being submitted, or is active.

Mandates can be resubmitted up to 10 times.

Relative endpoint: POST /mandates/MD123/actions/reinstate

**Restricted:** this endpoint is [restricted](#) to accounts with approved payment pages. To instead use the GoCardless hosted payment pages, see the [redirect flows](#) endpoint.

**Warning:** A mandate can only be reinstated if it has been cancelled through the dashboard or the API. Any mandate that has been cancelled by the bank cannot be reinstated.

### Parameters

## metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

HTTP PHP Python Ruby Java Java  
HTTP

```
POST https://api.gocardless.com  
Content-Type: application/json  
{  
  "data": {  
    "metadata": {  
      "ticket_id": "TK123"  
    }  
  }  
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
{
  "mandates": {
    "id": "MD123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "reference": "REF-123",
    "status": "submitted",
    "scheme": "bacs",
    "next_possible_charge_date": "2014-11-10",
    "metadata": {
      "contract": "ABCD1234"
    },
    "links": {
      "customer_bank_account": "BA123",
      "creditor": "CR123",
      "customer": "CU123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->mandates()->reinstate("MD123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandates.reinstate("MD123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.mandates.reinstate("MD123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.mandates().reinstate("MD123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const mandateResponse = await client.mandates.reinstate("MD123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var mandateResponse = await gocardless.Mandates.ReinstateAsync("MD0123");
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  mandateReinstateParams := go
  mandate, err := client.Manda
}
```

## Mandate Imports

Mandate Imports allow you to mi

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

The process is as follows:



Consultez notre politique de confidentialité

1. [Create a mandate import](#)

2. [Add entries](#) to the import
3. [Submit](#) the import
4. Wait until a member of the GoCardless team approves the import, at which point the mandates will be created
5. [Link up the mandates](#) in your system

When you add entries to your mandate import, they are not turned into actual mandates until the mandate import is submitted by you via the API, and then processed by a member of the GoCardless team. When that happens, a mandate will be created for each entry in the import.

We will issue a `mandate_created` webhook for each entry, which will be the same as the webhooks triggered when [creating a mandate](#) using the mandates API. Once these webhooks start arriving, any reconciliation can now be accomplished by [checking the current status](#) of the mandate import and [linking up the mandates to your system](#).

Note that all Mandate Imports have an upper limit of 30,000 entries, so we recommend you split your import into several smaller imports if you're planning to exceed this threshold.

**Restricted:** This API is currently only available for approved integrators - please [get in touch](#) if you would like to use this API.

## Properties

- `id`  
Unique identifier, beginning with "IM".
- `created_at`  
Fixed [timestamp](#), recording when this resource was created.
- `scheme`  
The scheme of the mandates to be imported.  
All mandates in a single mandate import must be for the same scheme.
- `status`  
The status of the mandate import.
  - `created`: A new mandate import.
  - `submitted`: After the integrator has finished adding mandates and [submitted](#) the import.
  - `cancelled`: If the integrator [cancelled](#) the mandate import.
  - `processing`: Once a mandate import has been approved by a GoCardless team member it will be in this state while mandates are imported.
  - `processed`: When all mandates have been imported successfully.
- `links[creditor]`  
ID of the associated creditor.

## Create a new mandate import

Mandate imports are first created, before mandates are added one-at-a-time, so this endpoint merely signals the start of the import process. Once you've finished adding entries to an import, you should [submit](#) it.

Relative endpoint: POST /mandate\_imports

### Parameters

- `scheme`  
*required* A bank payment scheme. Currently "ach", "autogiro", "bacs", "becs", "becs\_nz", "betalingsservice", "faster\_payments", "pad", "pay\_to" and "sepa\_core" are supported.
- `links[creditor]`  
ID of the associated creditor. Only required if your account manages multiple creditors.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/mandate_imports HTTP/1.1
Content-Type: application/json
{
  "mandate_imports": {
    "scheme": "bacs"
  }
}
```

```
HTTP/1.1 201 Created
Location: /mandate_imports/IM000010790WX1
Content-Type: application/json
{
```

```
  "mandate_imports": {
    "id": "IM000010790WX1",
    "scheme": "bacs",
    "status": "created",
    "created_at": "2018-03-12T10:00:00Z",
    "links": {
      "creditor": "CR123"
    }
  }
}
```

### PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token',
'environment' => \GoCardlessPro\Environment::SANDBOX);
]);

$client->mandateImports()->create([
  'params' => ['scheme' => 'bacs']
]);
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandate_imports.create(params={
    "scheme": "bacs"
})
```

Ruby

```
@client = GoCardlessPro::Client.new(  
  access_token: "your_access_token",  
  environment: :sandbox  
)
```

client = GoCardlessPro::Client.create(  
 access\_token: "your\_access\_token",  
 environment: :sandbox  
)

f)

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

MandateImport import_ = client.mandateImports().create()
    .withScheme(com.gocardless.services.MandateImportService.MandateImportCreateRequest.Scheme.BACS)
    .execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants Environments.Sandbox)
```

const

```
});  
  
.NET  
  
String accessToken = "your_access_token";  
GoCardlessClient goCardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);  
  
var importRequest = new GoCardless.Services.MandateImportCreateRequest()  
{  
    Scheme = MandateImportCreateRequest.MandateImportScheme.Bacs  
};  
  
var importResponse = await goCardless.MandateImports.CreateAsync(importRequest);  
GoCardless.Resources.MandateImport import_importResponse = importResponse.MandateImports[0];
```

G-8

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    mandateImportCreateParams := gocardless.MandateImportCreateParams{
        Scheme: "bacs",
    }

    mandateImport, err := client.MandateImports.Create(context, mandateImportCreateParams)
}
```

### Get a mandate import

Returns a single mandate import.

Relative endpoint: GET /mandate

## Parameters

                                                                                                                                                                                   <img alt="white circle icon" data-bbox="11715 890 11733 920

```
GET https://api.gocardless.com/  
HTTP/1.1 200 OK  
Content-Type: application/json
```

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"mandate_imports": {
  "id": "IM000010790WX1",
  "scheme": "bacs",
  "status": "created",
  "created_at": "2018-03-12T14:03:04.000Z",
  "links": {
    "creditor": "CR123"
  }
}

```

PHP

```

$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

```

```
$client->mandateImports()->get("IM000010790WX1");
```

#### Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandate_imports.get("IM000010790WX1")

```

#### Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.mandate_imports.get("IM000010790WX1")

```

#### Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

MandateImport import = client.mandateImports().get("IM000010790WX1").execute();

```

#### JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const mandateImport = await client.mandateImports.find("IM000010790WX1");

```

#### .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var response = await gocardless.MandateImports.GetAsync("IM000010790WX1");
GoCardless.Resources.MandateImport import = response.MandateImport;

```

#### Go

```

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  mandateImport, err := client.MandateImports.Get("IM000010790WX1")
}

```

### Submit a mandate import

Submits the mandate import, which is added to it.

In our sandbox environment, to aid you to test both the “submitted” response

Relative endpoint: POST /mandate\_imports

#### Parameters

HTTP PHP Python Ruby Java JavaScript

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

tted, it can no longer have entries

e submitted. This will allow you

## HTTP

```
POST https://api.gocardless.com/mandate_imports/IM000010790WX1/actions/submit HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "mandate_imports": {
    "id": "IM000010790WX1",
    "scheme": "bacs",
    "status": "submitted",
    "created_at": "2018-03-12T14:03:04.000Z",
    "links": {
      "creditor": "CR123"
    }
  }
}
```

```
POST https://api-sandbox.gocardless.com/mandate_imports/IM000010790WX1/actions/submit HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "mandate_imports": {
    "id": "IM000010790WX1",
    "scheme": "bacs",
    "status": "processing",
    "created_at": "2018-03-12T14:03:04.000Z",
    "links": {
      "creditor": "CR123"
    }
  }
}
```

## PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
$client->mandateImports()->submit("IM000010790WX1");
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
client.mandate_imports.submit("IM000010790WX1")
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.mandate_imports.submit("IM000010790WX1")
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
client.mandateImports().submit("IM000010790WX1").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const mandateImport = await client.mandateImports.submit("IM000010790WX1");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var response = await gocardless.MandateImports.SubmitAsync("IM000010790WX1");
GoCardless.Resources.MandateImport import = response.MandateImport;
```

## Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in initialising client: %v", err.Error())
    return
  }
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

```

}

mandateImportSubmitParams := map[string]interface{}
mandateImport, err := client.MandateImports.Submit(context, "IM000010790WX1", mandateImportSubmitParams)
}

```

## Cancel a mandate import

Cancels the mandate import, which aborts the import process and stops the mandates being set up in GoCardless. Once the import has been cancelled, it can no longer have entries added to it. Mandate imports which have already been submitted or processed cannot be cancelled.

Relative endpoint: POST /mandate\_imports/IM000010790WX1/actions/cancel

Parameters

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com/mandate\_imports/IM000010790WX1/actions/cancel HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "mandate_imports": {
    "id": "IM000010790WX1",
    "scheme": "bacs",
    "status": "cancelled",
    "created_at": "2018-03-12T14:03:04.000Z",
    "links": {
      "creditor": "CR123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->mandateImports()->cancel("IM000010790WX1");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandate_imports.cancel("IM000010790WX1")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.mandate_imports.cancel("IM000010790WX1")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.mandateImports().cancel("IM000010790WX1").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const mandateImportResponse = await client.mandateImports.cancel("IM000010790WX1");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var response = await gocardless.MandateImports.Cancel("IM000010790WX1");
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }

  client := gocardless.NewClient(config)
  _, err = client.MandateImports.Cancel("IM000010790WX1")
  if err != nil {
    fmt.Printf("got err in cancel: %v\n", err)
    return
  }
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

}
client, err := gocardless.New(config)
if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
}

mandateImportCancelParams := make(map[string]interface{})
mandateImport, err := client.MandateImports.Cancel(context, "IM000010790WX1", mandateImportCancelParams)

}

```

## Mandate Import Entries

Mandate Import Entries are added to a [Mandate Import](#). Each entry corresponds to one mandate to be imported into GoCardless.

To import a mandate you will need:

1. Identifying information about the customer (name/company and address)
2. Bank account details, consisting of an account holder name and either an IBAN or [local bank details](#)
3. Amendment details (SEPA only)

We suggest you provide a `record_identifier` (which is unique within the context of a single mandate import) to help you to identify mandates that have been created once the import has been processed by GoCardless. You can [list the mandate import entries](#), match them up in your system using the `record_identifier`, and look at the `links` fields to find the mandate, customer and customer bank account that have been imported.

**Restricted:** This API is currently only available for approved integrators - please [get in touch](#) if you would like to use this API.

Properties

```

created_at
    Fixed timestamp, recording when this resource was created.
processing_errors
    Per-resource processing errors
record_identifier
    A unique identifier for this entry, which you can use (once the import has been processed by GoCardless) to identify the records that have been created.
    Limited to 255 characters.
links[customer]
    The ID of the customer which was created when the mandate import was processed.
links[customer_bank_account]
    The ID of the customer bank account which was created when the mandate import was processed.
links[mandate]
    The ID of the mandate which was created when the mandate import was processed.
links[mandate_import]
    The ID of the mandate import. This is returned when you create a Mandate Import.

```

## Add a mandate import entry

For an existing [mandate import](#), this endpoint can be used to add individual mandates to be imported into GoCardless.

You can add no more than 30,000 rows to a single mandate import. If you attempt to go over this limit, the API will return a `record_limit_exceeded` error.

Relative endpoint: POST /mandate\_import\_entries

Parameters

```

bank_account[account_holder_name]
    required Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required
    unless the request includes a customer bank account token.
bank_account[account_number]
    Bank account number - see local details for more information. Alternatively you can provide an iban.
bank_account[account_type]
    Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See local details for more
    information.
bank_account[bank_code]
    Bank code - see local details for more information. Alternatively you can provide an iban.
bank_account[branch_code]
    Branch code - see local details for more information. Alternatively you can provide an iban.
bank_account[country_code]
    ISO 3166-1 alpha-2 code. Defaults to the country code of the iban if supplied, otherwise is required.
bank_account[iban]
    International Bank Account Number. Alternatively you can provide local details. IBANs are not accepted for Swedish bank accounts denominated in SEK -
    you must supply local details.
bank_account[metadata]
    Key-value store of custom data.
customer[address_line1]
    The first line of the customer's address.
customer[address_line2]
    The second line of the customer's address.
customer[address_line3]
    The third line of the customer's address.
customer[city]
    The city of the customer's address.
customer[company_name]
    Customer's company name. Note that any mandate created from this API will be considered to be a "Personal PAD").
customer[country_code]
    ISO 3166-1 alpha-2 code.
customer[danish_identity_number]

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

a `company_name` value will mean  
considered to be a "Personal PAD").



Consultez notre politique de confidentialité

For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer's bank account is denominated in Danish krone (DKK).

**customer[email]**

Customer's email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

**customer[family\_name]**

Customer's surname. Required unless a `company_name` is provided.

**customer[given\_name]**

Customer's first name. Required unless a `company_name` is provided.

**customer[language]**

[ISO 639-1](#) code. Used as the language for notification emails sent by GoCardless if your organisation does not send its own (see [compliance requirements](#)).

Currently only "en", "fr", "de", "pt", "es", "it", "nl", "da", "nb", "sl", "sv" are supported. If this is not provided, the language will be chosen based on the `country_code` (if supplied) or default to "en".

**customer[metadata]**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**customer[phone\_number]**

[ITU E.123](#) formatted phone number, including country code.

**customer[postal\_code]**

The customer's postal code. Required if mandate import scheme is either `bacs` or `sepa`.

**customer[region]**

The customer's address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

**customer[swedish\_identity\_number]**

For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer's bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.

**amendment[original\_creditor\_id]**

The creditor identifier of the direct debit originator. Required if mandate import scheme is `sepa`.

**amendment[original\_creditor\_name]**

Data about the original mandate to be moved or modified.

**amendment[original\_mandate\_reference]**

The unique SEPA reference for the mandate being amended. Required if mandate import scheme is `sepa`.

**mandate[metadata]**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**mandate[reference]**

Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

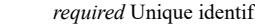
**record\_identifier**

A unique identifier for this entry, which you can use (once the import has been processed by GoCardless) to identify the records that have been created.

Limited to 255 characters.

**links[mandate\_import]**

*required* Unique identifier, beginning with "IM".



HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/mandate_import_entries HTTP/1.1
Content-Type: application/json
{
  "mandate_import_entries": {
    "links": {
      "mandate_import": "IM000010790WX1"
    },
    "record_identifier": "bank-file.xml/line-1",
    "customer": {
      "company_name": "Jane's widgets",
      "email": "jane@janeswidgets.fr"
    },
    "bank_account": {
      "account_holder_name": "Jane Doe",
      "iban": "FR14BARC20000055779911"
    },
    "amendment": {
      "original_mandate_reference": "REFNMANDATE",
      "original_creditor_id": "FR1230THERBANK",
      "original_creditor_name": "Existing DD Provider"
    }
  }
}
```

HTTP/1.1 201 Created

Content-Type: application/json

```
{
  "mandate_import_entries": {
    "record_identifier": "bank-file.xml/line-1",
    "created_at": "2018-03-03T00:00:00Z",
    "links": {
      "mandate_import": "IM000010790WX1"
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client;
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment::TEST;
]);
$client->mandateImportEntries(
  "params" => [
    "links" => [
      "mandate_import" => "IM000010790WX1"
    ],
    "record_identifier" => "bank-file.xml/line-1",
    "customer" => [
      "company_name" => "Jane's widgets",
      "email" => "jane@janeswidgets.fr"
    ],
  ]
);
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"bank_account" => [
  "account_holder_name" => "Jane Doe",
  "iban" => "FR14BARC20000055779911"
],
"amendment" => [
  "original_mandate_reference" => "REFNMANDATE",
  "original_creditor_id" => "FR1230THERBANK",
  "original_creditor_name" => "Existing DD Provider"
]
]
]);
Python

```

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandate_import_entries.create(params={
    "links": {
        "mandate_import": "IM000010790WX1"
    },
    "record_identifier": "bank-file.xml/line-1",
    "customer": {
        "company_name": "Jane's widgets",
        "email": "jane@janeswidgets.fr"
    },
    "bank_account": {
        "account_holder_name": "Jane Doe",
        "iban": "FR14BARC20000055779911"
    },
    "amendment": {
        "original_mandate_reference": "REFNMANDATE",
        "original_creditor_id": "FR1230THERBANK",
        "original_creditor_name": "Existing DD Provider"
    }
})

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.mandate_import_entries.create(params: {
  links: {
    mandate_import: "IM000010790WX1"
  },
  record_identifier: "bank-file.xml/line-1",
  customer: {
    company_name: "Jane's widgets",
    email: "jane@janeswidgets.fr"
  },
  bank_account: {
    account_holder_name: "Jane Doe",
    iban: "FR14BARC20000055779911"
  },
  amendment: {
    original_mandate_reference: "REFNMANDATE",
    original_creditor_id: "FR1230THERBANK",
    original_creditor_name: "Existing DD Provider"
  }
})

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

MandateImport import = client.mandateImportEntries().create()
    .withCustomerCompanyName("Jane's widgets")
    .withCustomerEmail("jane@janeswidgets.fr")
    .withBankAccountAccountHolderName("Jane Doe")
    .withBankAccountIban("FR14BARC20000055779911")
    .withAmendmentOriginalMandateReference("REFNMANDATE")
    .withAmendmentOriginalCreditorId("FR1230THERBANK")
    .withAmendmentOriginalCreditorName("Existing DD Provider")
    .withLinksMandateImport("IM000010790WX1")
    .execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your

const mandateImportEntry = await
  links: {
    mandate_import: "IM000010790WX1"
  },
  record_identifier: "bank-file.xml/line-1",
  customer: {
    company_name: "Jane's widgets",
    email: "jane@janeswidgets.fr"
  },
  bank_account: {
    account_holder_name: "Jane Doe",
    iban: "FR14BARC20000055779911"
  },
  amendment: {
    original_mandate_reference: "REFNMANDATE",
    original_creditor_id: "FR1230THERBANK",
    original_creditor_name: "Existing DD Provider"
  }
}

```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        original_creditor_id: "FR1230THERBANK",
        original_creditor_name: "Existing DD Provider"
    }
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var request = new GoCardless.Services.MandateImportEntryCreateRequest()
{
    Customer = new GoCardless.Services.MandateImportEntryCreateRequest.MandateImportEntryCustomer()
    {
        CompanyName = "Jane's widgets"
        Email = "jane@janewidgets.fr"
    }
    BankAccount = new GoCardless.Services.MandateImportEntryCreateRequest.MandateImportEntryBankAccount()
    {
        AccountHolderName = "Jane Doe"
        Iban = "FR14BARC20000055779911"
    }
    Amendment = new GoCardless.Services.MandateImportEntryCreateRequest.MandateImportEntryAmendment()
    {
        OriginalMandateReference = "REFMANDATE"
        OriginalCreditorId = "FR1230THERBANK"
        OriginalCreditorName = "Existing DD Provider"
    }
    Links = new GoCardless.Services.MandateImportEntryCreateRequest.MandateImportEntryLinks()
    {
        MandateImport = "IM000010790WX1"
    }
};

var importResponse = await gocardless.MandateImportEntries.CreateAsync(request);
GoCardless.Resources.MandateImportEntry entry = importResponse.MandateImportEntry;

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    mandateImportEntryCreateParams := gocardless.MandateImportEntryCreateParams{
        Customer: gocardless.MandateImportEntryCreateParamsCustomer{
            CompanyName: "Théâtre du Palais-Royal",
            Email: "moliere@tdpr.fr",
        },
        BankAccount: gocardless.MandateImportEntryCreateParamsBankAccount{
            AccountHolderName: "Jean-Baptiste Poquelin",
            Iban: "FR14BARC20000055779911",
        },
        Amendment: &gocardless.MandateImportEntryCreateParamsAmendment{
            OriginalMandateReference: "REFMANDATE",
            OriginalCreditorId: "FR1230THERBANK",
            OriginalCreditorName: "Amphitryon",
        },
        Links: gocardless.MandateImportEntryCreateParamsLinks{
            MandateImport: "IM000010790WX1",
        },
    }
}

mandateImportEntry, err := client.MandateImportEntries.Create(context, mandateImportEntryCreateParams)
}

```

## List all mandate import entries

For an existing mandate import, this endpoint lists all of the entries attached.

After a mandate import has been submitted, you can use this endpoint to associate records in your system (using the `record_identifier` that you provided when creating the mandate import).

Relative endpoint: GET `/mandate_import/{mandate_import}/entries`

Parameters

`mandate_import`  
`required Unique identifier,`  
`after`  
`Cursor pointing to the start`  
`before`  
`Cursor pointing to the end`  
`limit`  
`Number of records to return`  
`status`

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

One of:

- `successfully_processed`: the entry has been imported and the associated records created.
- `unsuccessfully_processed`: the entry could not be processed due to an error, see the '`processing_errors`' value

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET [https://api.gocardless.com/mandate\\_import\\_entries?mandate\\_import=IM000010790WX1](https://api.gocardless.com/mandate_import_entries?mandate_import=IM000010790WX1) HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "mandate_import_entries": [
    {
      "record_identifier": "bank-file.xml/line-2",
      "created_at": "2018-03-03T00:00:01Z",
      "links": {
        "mandate_import": "IM000010790WX1"
      },
      "processing_errors": null
    },
    {
      "record_identifier": "bank-file.xml/line-1",
      "created_at": "2018-03-03T00:00:00Z",
      "links": {
        "mandate_import": "IM000010790WX1"
      },
      "processing_errors": {
        "mandate": ["Reference matches the start of the reference used by another mandate"]
      }
    }
  ],
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->mandateImportEntries()->all([
  'params' => ['mandate_import' => "IM000010790WX1"]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandate_import_entries.all(
  params= { "mandate_import": "IM000010790WX1" }
).records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.mandate_import_entries.all(
  params: {
    "mandate_import" => "IM000010790WX1"
  }
).each { |entry| puts entry.record_identifier }
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

for (MandateImportEntry entry : client.mandateImportEntries().all().withMandateImport("IM000010790WX1").execute())
  System.out.println(entry.get());
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');
const mandateImportEntries = await client.mandateImportEntries();
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient
  .newBuilder()
  .withAccessToken(accessToken)
  .withEnvironment(Environment.SANDBOX)
  .build();

var request = new GoCardless.Services.MandateImportEntryListRequest()
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Consultez notre politique de confidentialité



```
{
  "id": "NBL123",
  "created_at": "2014-05-08T17:01:06.000Z",
  "balance_limit": "10000",
  "currency": "GBP",
  "links": {
    "creditor": "CR123"
  }
}
]
```

**PHP**

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->negativeBalanceLimits()->list([
  'params' => [
    'currency' => "GBP",
    'creditor' => "CR123",
  ]
]);
```

**Python**

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.negative_balance_limits.list(params={
  'currency': "GBP",
  'creditor': "CR123",
}).records
```

**Ruby**

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.negative_balance_limits.list(
  params: {
    currency: "GBP",
    creditor: "CR123",
  }
).records
```

**Java**

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.negativeBalanceLimits().
  all().
  withCreditor("CR123").
  withCurrency("GBP").
  execute();
```

**JavaScript**

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

await client.negativeBalanceLimits.list({
  currency: "GBP",
  creditor: "CR123",
});
```

**.NET**

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var request = new GoCardless.Services.NegativeBalanceLimitListRequest()
{
  Currency = "GBP",
  Creditor = "CR123",
};

gocardless.NegativeBalanceLimit
```

**Go**

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.New(config)
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
}

params := gocardless.NegativeBalanceLimitListParams{
    Currency: "GBP",
    Creditor: "CR123",
}
response, err := client.NegativeBalanceLimits.List(context, params)
}

```

## Outbound Payments

Outbound Payments represent payments sent from [creditors](#).

GoCardless will notify you via a [webhook](#) when the status of the outbound payment [changes](#).

**Restricted:** Outbound Payments are currently in Early Access and available only to a limited list of organisations. If you are interested in using this feature, please stay tuned for our public launch announcement. We are actively testing and refining our API to ensure it meets your needs and provides the best experience.

### Properties

**id** Unique identifier of the outbound payment.  
**amount** Amount, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).  
**created\_at** Fixed [timestamp](#), recording when the outbound payment was created.  
**currency** [ISO 4217](#) currency. Currently only “GBP” is supported.  
**description** A human-readable description of the outbound payment  
**execution\_date** A future date on which the outbound payment should be sent. If not specified, the payment will be sent as soon as possible.  
**is\_withdrawal** Indicates whether the outbound payment is a withdrawal to your verified business bank account.  
**metadata** Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.  
**reference** An optional reference that will appear on your customer’s bank statement. The character limit for this reference is dependent on the scheme.  
**Faster Payments** - 18 characters, including: “ABCDEFHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 &-/”  
**scheme** Bank payment scheme to process the outbound payment. Currently only “faster\_payments” (GBP) is supported.  
**status** One of:

- **verifying**: The payment has been [created](#) and the verification process has begun.
- **pending\_approval**: The payment is awaiting [approval](#).
- **scheduled**: The payment has passed verification & [approval](#), but processing has not yet begun.
- **executing**: The execution date has arrived and the payment has been placed in queue for processing.
- **executed**: The payment has been accepted by the scheme and is now on its way to the recipient.
- **cancelled**: The payment has been [cancelled](#) or was not [approved](#) on time.
- **failed**: The payment was not sent, usually due to an error while or after executing.

**recipient\_bank\_account\_holder\_verification[actual\_account\_name]**

- The actual account name returned by the recipient’s bank, populated only in the case of a partial match.

**recipient\_bank\_account\_holder\_verification[result]**
Result of the verification, could be one of

- **full\_match**: The verification has confirmed that the account name exactly matches the details provided.
- **partial\_match**: The verification has confirmed that the account name is similar but does not match to the details provided.
- **no\_match**: The verification concludes the provided name does not match the account details.
- **unable\_to\_match**: The verification could not be performed due to recipient bank issues or technical issues

**recipient\_bank\_account\_holder\_**  
Type of the verification that  
**links[creditor]** ID of the creditor who send  
**links[customer]** ID of the [customer](#) that receive  
**links[recipient\_bank\_account]** ID of the customer bank acc

## Create an outbound payment

Relative endpoint: POST /outboundpayments

### Parameters

<https://developer.gocardless.com/api-reference/>

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**amount**

*required* Amount, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**scheme**

*required* Bank payment scheme to process the outbound payment. Currently only “faster\_payments” (GBP) is supported.

**description**

A human-readable description of the outbound payment

**execution\_date**

A future date on which the outbound payment should be sent. If not specified, the payment will be sent as soon as possible.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**reference**

An optional reference that will appear on your customer’s bank statement. The character limit for this reference is dependent on the scheme.

**Faster Payments** - 18 characters, including: “ABCDEFHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 &-./”

**links[creditor]**

ID of the creditor who sends the outbound payment.

**links[recipient\_bank\_account]**

*required* ID of the customer bank account which receives the outbound payment.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/outbound_payments HTTP/1.1
```

Content-Type: application/json

```
{
  "outbound_payments": {
    "amount": 10,
    "scheme": "faster_payments",
    "description": "Reward Payment (August 2024)",
    "links": {
      "recipient_bank_account": "BA12343",
      "creditor": "CR123"
    }
  }
}
```

HTTP/1.1 201 Created

Content-Type: application/json

```
{
  "outbound_payments": {
    "id": "OUT01JR7P5PKW3K7Q34CJAWC03E82",
    "created_at": "2024-09-05T12:20:04.397Z",
    "status": "pending_approval",
    "amount": 10,
    "scheme": "faster_payments",
    "currency": "GBP",
    "description": "Reward Payment (August 2024)",
    "execution_date": "2024-09-05",
    "reference": "GC-QC2FI7GBEW7VCXL",
    "is_withdrawal": false,
    "links": {
      "creditor": "CR123",
      "recipient_bank_account": "BA123"
    },
    "verifications": {
      "recipient_account_holder_verification": {
        "result": "nil",
        "type": "confirmation_of_payee",
        "actual_account_name": null
      }
    },
    "metadata": null
  }
}
```

PHP

Python

Ruby

Java

JavaScript

.NET

Go

**Create a withdrawal outbound**

Creates an outbound payment to your bank account.

Relative endpoint: POST /outbound\_payments

Parameters

**Nous utilisons des cookies**

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**amount**  
*required* Amount, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**scheme**  
*required* Bank payment scheme to process the outbound payment. Currently only “faster\_payments” (GBP) is supported.

**description**  
A human-readable description of the outbound payment

**execution\_date**  
A future date on which the outbound payment should be sent. If not specified, the payment will be sent as soon as possible.

**metadata**  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**reference**  
An optional reference that will appear on your customer’s bank statement. The character limit for this reference is dependent on the scheme.  
**Faster Payments** - 18 characters, including: “ABCDEFHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 &.-/”

**links[creditor]**  
ID of the creditor who sends the outbound payment.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/outbound_payments/withdrawal HTTP/1.1
Content-Type: application/json
{
  "outbound_payments": {
    "amount": 10,
    "scheme": "faster_payments",
    "description": "Withdraw funds to business account."
  }
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json
{
  "outbound_payments": {
    "id": "OUT01JR7P5PKW3K7Q34CJAWC03E82",
    "created_at": "2024-09-05T12:20:04.397Z",
    "status": "pending_approval",
    "amount": 10,
    "scheme": "faster_payments",
    "currency": "GBP",
    "description": "Withdraw funds to business account.",
    "execution_date": "2024-09-05",
    "reference": "GC-QC2FI7GBEW7VCXL",
    "is_withdrawal": true,
    "verifications": {
      "recipient_account_holder_verification": null
    },
    "links": {
      "creditor": "CR123",
      "recipient_bank_account": "BA123"
    },
    "metadata": null
  }
}
```

PHP

Python

Ruby

Java

JavaScript

.NET

Go

## Cancel an outbound payment

Cancels an outbound payment. Once an outbound payment is executing, the money moving process cannot be stopped.

Relative endpoint: POST /outbound\_payments/{id}/cancel

Parameters

**metadata**

Key-value store of custom data.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

led. Once an outbound payment

rs.

```
POST https://api.gocardless.com/outbound_payments/OUT01JR7P5PKW3K7Q34CJAWC03E82/actions/cancel
```

```
HTTP/1.1 200
Content-Type: application/json
{
  "outbound_payments": {
    "id": "OUT01JR7P5PKW3K7Q34CJAWC03E82",
    "created_at": "2024-09-05T12:20:04.397Z",
    "status": "cancelled",
    "amount": 10,
    "scheme": "faster_payments",
    "currency": "GBP",
    "description": "Reward Payment (August 2024)",
    "execution_date": "2024-09-05",
    "reference": "GC-QC2FI7GBEW7VCXL",
    "is_withdrawal": false,
    "links": {
      "creditor": "CR123",
      "recipient_bank_account": "BA123"
    },
    "verifications": {
      "recipient_account_holder_verification": {
        "result": "FULL_MATCH",
        "type": "confirmation_of_payee",
        "actual_account_name": null
      }
    },
    "metadata": null
  }
}
PHP
```

Python

Ruby

Java

JavaScript

.NET

Go

## Approve an outbound payment

Approves an outbound payment. Only outbound payments with the “pending\_approval” status can be approved.

Relative endpoint: POST /outbound\_payments/OUT01JR7P5PKW3K7Q34CJAWC03E82/actions/approve

Parameters



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/outbound_payments/OUT01JR7P5PKW3K7Q34CJAWC03E82/actions/approve
```

```
HTTP/1.1 200
Content-Type: application/json
{
```

```
  "outbound_payments": {
    "id": "OUT01JR7P5PKW3K7Q34CJAWC03E82",
    "created_at": "2024-09-05T12:20:04.397Z",
    "status": "scheduled",
    "amount": 10,
    "scheme": "faster_payments",
    "currency": "GBP",
    "description": "Reward Payment (August 2024)",
    "execution_date": "2024-09-05",
    "reference": "GC-QC2FI7GBEW7VCXL",
    "is_withdrawal": false,
    "links": {
      "creditor": "CR123",
      "recipient_bank_account": "BA123"
    },
    "verifications": {
      "recipient_account_holder_verification": {
        "result": "FULL_MATCH",
        "type": "confirmation_of_payee",
        "actual_account_name": null
      }
    },
    "metadata": null
  }
}
PHP
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

[Python](#)[Ruby](#)[Java](#)[JavaScript](#)[.NET](#)[Go](#)

## Get an outbound payment

Fetches an outbound\_payment by ID

Relative endpoint: GET /outbound\_payments/OUT01JR7P5PKW3K7Q34CJAWC03E82



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/outbound_payments/OUT01JR7P5PKW3K7Q34CJAWC03E82
```

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "outbound_payments": {
    "id": "OUT01JR7P5PKW3K7Q34CJAWC03E82",
    "created_at": "2024-09-05T12:20:04.397Z",
    "status": "scheduled",
    "amount": 10,
    "scheme": "faster_payments",
    "currency": "GBP",
    "description": "Reward Payment (August 2024)",
    "execution_date": "2024-09-05",
    "reference": "GC-QC2FI7GBEW7VCXL",
    "is_withdrawal": false,
    "links": {
      "creditor": "CR123",
      "recipient_bank_account": "BA123"
    },
    "verifications": {
      "recipient_account_holder_verification": {
        "result": "FULL_MATCH",
        "type": "confirmation_of_payee",
        "actual_account_name": null
      }
    },
    "metadata": null
  }
}
```

PHP

[Python](#)

[Ruby](#)

[Java](#)

[JavaScript](#)

[.NET](#)

[Go](#)

## List outbound payments

Returns a [cursor-paginated](#) list of

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Relative endpoint: GET /outbound\_



Consultez notre politique de confidentialité

Parameters

**after** Cursor pointing to the start of the desired set.

**before** Cursor pointing to the end of the desired set.

**created\_from** The beginning of query period

**created\_to** The end of query period

**limit** Number of records to return.

**status** One of:

- **verifying**: The payment has been [created](#) and the verification process has begun.
- **pending\_approval**: The payment is awaiting [approval](#).
- **scheduled**: The payment has passed verification & [approval](#), but processing has not yet begun.
- **executing**: The execution date has arrived and the payment has been placed in queue for processing.
- **executed**: The payment has been accepted by the scheme and is now on its way to the recipient.
- **cancelled**: The payment has been [cancelled](#) or was not [approved](#) on time.
- **failed**: The payment was not sent, usually due to an error while or after executing.

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET [https://api.gocardless.com/outbound\\_payments](https://api.gocardless.com/outbound_payments) HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "outbound_payments": [
    {
      "id": "OUT01JR7P5PKW3K7Q34CJAWC03E82",
      "created_at": "2024-09-05T12:20:04.397Z",
      "status": "cancelled",
      "amount": 10,
      "scheme": "faster_payments",
      "currency": "GBP",
      "description": "Reward Payment (August 2024)",
      "execution_date": "2024-09-05",
      "reference": "GC-QC2FI7GBEW7VCXL",
      "is_withdrawal": false,
      "links": {
        "creditor": "CR123",
        "recipient_bank_account": "BA123"
      },
      "verifications": {
        "recipient_account_holder_verification": {
          "result": "FULL_MATCH",
          "type": "confirmation_of_payee",
          "actual_account_name": null
        }
      },
      "metadata": null
    }
  ]
}
```

PHP

Python

Ruby

Java

JavaScript

.NET

Go

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

### Update an outbound payment



Consultez notre politique de confidentialité

Updates an outbound payment object. This accepts only the metadata parameter.

Relative endpoint: PUT /outbound\_payments/OUT01JR7P5PKW3K7Q34CJAWC03E82

#### Parameters

##### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

PUT https://api.gocardless.com/outbound\_payments/OUT01JR7P5PKW3K7Q34CJAWC03E82

Content-Type: application/json

```
{
  "outbound_payments": {
    "metadata": {
      "id": "1234"
    }
  }
}
```

HTTP/1.1 200

Content-Type: application/json

```
{
  "outbound_payments": {
    "id": "OUT01JR7P5PKW3K7Q34CJAWC03E82",
    "created_at": "2024-09-05T12:20:04.397Z",
    "status": "cancelled",
    "amount": 10,
    "scheme": "faster_payments",
    "currency": "GBP",
    "description": "Reward Payment (August 2024)",
    "execution_date": "2024-09-05",
    "reference": "GC-QC2FI7GBEW7VCXL",
    "is_withdrawal": false,
    "links": {
      "creditor": "CR123",
      "recipient_bank_account": "BA123"
    },
    "verifications": {
      "recipient_account_holder_verification": {
        "result": "FULL_MATCH",
        "type": "confirmation_of_payee",
        "actual_account_name": null
      }
    },
    "metadata": {
      "id": "1234"
    }
  }
}
```

PHP

Python

Ruby

Java

JavaScript

.NET

Go

## Payer Authorisations

Don't use Payer Authorisations for new integrations. It is deprecated in favour of [Billing Requests](#). Use Billing Requests to build any future integrations.

Payer Authorisation resource acts as a wrapper for creating customer bank account and mandate details in a single request. PayerAuthorisation API enables the integrators to build their own cust

The process to use the Payer Auth

1. Create a Payer Authorisation
2. Update the authorisation with the mandate
3. Submit the authorisation, after which it becomes active
4. [coming soon] Redirect the user to a confirmation page
5. Confirm the authorisation to complete the process

After the Payer Authorisation is created, you can

To retrieve the status and ID of the authorisation, you can

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

ced as a non-breaking change)

 Consultez notre politique de confidentialité

1. Listen to `payer_authorisation_completed` [webhook](#) (recommended)

2. Poll the GET [endpoint](#)

3. Poll the GET events API [https://api.gocardless.com/events?payer\\_authorisation={id}&action=completed](https://api.gocardless.com/events?payer_authorisation={id}&action=completed)

Note that the `create` and `update` endpoints behave differently than other existing `create` and `update` endpoints. The Payer Authorisation is still saved if incomplete data is provided. We return the list of incomplete data in the `incomplete_fields` along with the resources in the body of the response. The bank account details(`account_number`, `bank_code` & `branch_code`) must be sent together rather than splitting across different request for both `create` and `update` endpoints.

The API is designed to be flexible and allows you to collect information in multiple steps without storing any sensitive data in the browser or in your servers.

## Properties

`id`

Unique identifier, beginning with “PA”.

`bank_account[account_holder_name]`

Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a [customer bank account token](#).

`bank_account[account_number]`

Bank account number - see [local details](#) for more information. Alternatively you can provide an `iban`.

`bank_account[account_number-ending]`

The last few digits of the account number. Currently 4 digits for NZD bank accounts and 2 digits for other currencies.

`bank_account[account_number_suffix]`

Account number suffix (only for bank accounts denominated in NZD) - see [local details](#) for more information.

`bank_account[account_type]`

Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.

`bank_account[bank_code]`

Bank code - see [local details](#) for more information. Alternatively you can provide an `iban`.

`bank_account[branch_code]`

Branch code - see [local details](#) for more information. Alternatively you can provide an `iban`.

`bank_account[country_code]`

[ISO 3166-1 alpha-2 code](#). Defaults to the country code of the `iban` if supplied, otherwise is required.

`bank_account[currency]`

[ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

`bank_account[iban]`

International Bank Account Number. Alternatively you can provide [local details](#). IBANs are not accepted for Swedish bank accounts denominated in SEK - you must supply [local details](#).

`bank_account[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`created_at`

[Timestamp](#), recording when this Payer Authorisation was created.

`customer[address_line1]`

The first line of the customer’s address.

`customer[address_line2]`

The second line of the customer’s address.

`customer[address_line3]`

The third line of the customer’s address.

`customer[city]`

The city of the customer’s address.

`customer[company_name]`

Customer’s company name. Required unless a `given_name` and `family_name` are provided. For Canadian customers, the use of a `company_name` value will mean that any mandate created from this customer will be considered to be a “Business PAD” (otherwise, any mandate will be considered to be a “Personal PAD”).

`customer[country_code]`

[ISO 3166-1 alpha-2 code](#).

`customer[danish_identity_number]`

For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer’s bank account is denominated in Danish krone (DKK).

`customer[email]`

Customer’s email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

`customer[family_name]`

Customer’s surname. Required unless a `company_name` is provided.

`customer[given_name]`

Customer’s first name. Required unless a `company_name` is provided.

`customer[locale]`

An [IETF Language Tag](#), used for both language and regional variations of our product.

`customer[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer[postal_code]`

The customer’s postal code.

`customer[region]`

The customer’s address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

`customer[swedish_identity_number]`

For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer’s bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.

`incomplete_fields`

An array of fields which are

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

rs.

For ACH customers only. Required for ACH customers. A string containing the IP address of the payer to whom the mandate belongs (i.e. as a result of their completion of a mandate setup flow in their browser).

Not required for creating offline mandates where `authorisation_source` is set to telephone or paper.

#### `mandate[reference]`

Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

#### `mandate[scheme]`

A bank payment scheme. Currently “ach”, “autogiro”, “baes”, “becs”, “becs\_nz”, “betalingsservice”, “faster\_payments”, “pad”, “pay\_to” and “sepa\_core” are supported.

#### `status`

One of:

- `created`: The PayerAuthorisation has been created, and not been confirmed yet
- `submitted`: The payer information has been submitted
- `confirmed`: PayerAuthorisation is confirmed and resources are ready to be created
- `completed`: The PayerAuthorisation has been completed and customer, `bank_account` and mandate has been created
- `failed`: The PayerAuthorisation has failed and customer, `bank_account` and mandate is not created

#### `links[bank_account]`

Unique identifier, beginning with “BA”.

#### `links[customer]`

Unique identifier, beginning with “CU”.

#### `links[mandate]`

Unique identifier, beginning with “MD”. Note that this prefix may not apply to mandates created before 2016.

## Get a single Payer Authorisation

Retrieves the details of a single existing Payer Authorisation. It can be used for polling the status of a Payer Authorisation.

Relative endpoint: GET /payer\_authorisations/PAU123



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/payer_authorisations/PAU123 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "payer_authorisations": {
    "id": "PAU123",
    "created_at": "2020-09-11T14:04:50.579Z",
    "customer": {
      "address_line1": "Unit 12, 2 Somerset Road",
      "address_line2": "terertertre",
      "city": "London",
      "company_name": "",
      "country_code": "US",
      "email": "mail@example.com",
      "family_name": "test",
      "given_name": "lastname",
      "postal_code": "1234",
      "region": "AR",
      "metadata": {}
    },
    "bank_account": {
      "account_holder_name": "test lastname",
      "account_numberEnding": "11",
      "account_type": "savings",
      "country_code": "US",
      "bank_name": "COMMUNITY FEDERAL SAVINGS BANK",
      "metadata": {}
    },
    "mandate": {
      "scheme": "ach",
      "metadata": {}
    },
    "status": "completed",
    "incomplete_fields": [],
    "links": {
      "customer": "CU00123",
      "bank_account": "BA00123",
      "mandate": "MD000123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment\Sandbox;
]);
$client->payerAuthorisations()
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
client.payer_authorisations.get("PAU123")
Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payer_authorisations.get("PAU123")

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

PayerAuthorisation payerAuthorisation = client.payerAuthorisations.get("PAU123").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payerAuthorisation = await client.payerAuthorisations.find("PAU123");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var payerAuthorisationResponse = await gocardless.payerAuthorisations.GetAsync("PAU123");
GoCardless.Resources.PayerAuthorisation payerAuthorisation = payerAuthorisationResponse.PayerAuthorisation;
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }
    payerAuthorisation, err := client.PayerAuthorisations.Get(context, "PAU123")
}
```

**Create a Payer Authorisation**

Creates a Payer Authorisation. The resource is saved to the database even if incomplete. An empty array of incomplete\_fields means that the resource is valid. The ID of the resource is used for the other actions. This endpoint has been designed this way so you do not need to save any payer data on your servers or the browser while still being able to implement a progressive solution, such as a multi-step form.

Relative endpoint: POST /payer\_authorisations

## Parameters

`bank_account[account_holder_name]`  
Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a [customer bank account token](#).

`bank_account[account_number]`  
Bank account number - see [local details](#) for more information. Alternatively you can provide an iban.

`bank_account[account_numberEnding]`  
The last few digits of the account number. Currently 4 digits for NZD bank accounts and 2 digits for other currencies.

`bank_account[account_number_suffix]`  
Account number suffix (only for bank accounts denominated in NZD) - see [local details](#) for more information.

`bank_account[account_type]`  
Bank account type. Requires information.

`bank_account[bank_code]`  
Bank code - see [local detail](#)

`bank_account[branch_code]`  
Branch code - see [local det](#)

`bank_account[country_code]`  
[ISO 3166-1 alpha-2 code](#). I

`bank_account[currency]`  
[ISO 4217 currency code](#). C

`bank_account[iban]`  
International Bank Account  
you must supply [local detai](#)

`bank_account[metadata]`

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

s. See [local details](#) for more



Consultez notre politique de confidentialité

counts denominated in SEK -

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer[address_line1]`  
The first line of the customer's address.

`customer[address_line2]`  
The second line of the customer's address.

`customer[address_line3]`  
The third line of the customer's address.

`customer[city]`  
The city of the customer's address.

`customer[company_name]`  
Customer's company name. Required unless a `given_name` and `family_name` are provided. For Canadian customers, the use of a `company_name` value will mean that any mandate created from this customer will be considered to be a "Business PAD" (otherwise, any mandate will be considered to be a "Personal PAD").

`customer[country_code]`  
[ISO 3166-1 alpha-2 code](#).

`customer[danish_identity_number]`  
For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer's bank account is denominated in Danish krone (DKK).

`customer[email]`  
Customer's email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

`customer[family_name]`  
Customer's surname. Required unless a `company_name` is provided.

`customer[given_name]`  
Customer's first name. Required unless a `company_name` is provided.

`customer[locale]`  
An [IETF Language Tag](#), used for both language and regional variations of our product.

`customer[metadata]`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer[postal_code]`  
The customer's postal code.

`customer[region]`  
The customer's address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

`customer[swedish_identity_number]`  
For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer's bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.

`mandate[metadata]`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`mandate[payer_ip_address]`  
For ACH customers only. Required for ACH customers. A string containing the IP address of the payer to whom the mandate belongs (i.e. as a result of their completion of a mandate setup flow in their browser).

Not required for creating offline mandates where `authorisation_source` is set to telephone or paper.

`mandate[reference]`  
Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

`mandate[scheme]`  
A bank payment scheme. Currently "ach", "autogiro", "bacs", "becs", "becs\_nz", "betalingsservice", "faster\_payments", "pad", "pay\_to" and "sepa\_core" are supported.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/payer_authorisations HTTP/1.1
Content-Type: application/json
{
  "payer_authorisations": {
    "customer": {
      "email": "mail@example.com",
      "given_name": "Name",
      "family_name": "Surname",
      "metadata": {
        "salesforce_id": "EFGH5678"
      }
    },
    "bank_account": {
      "account_holder_name": "Name Surname",
      "branch_code": "200000",
      "account_number": "55779911",
      "metadata": {}
    },
    "mandate": {
      "reference": "XYZ789",
      "metadata": {}
    }
  }
}
```

Example response with incomplete

```
HTTP/1.1 201 Created
Location: /payer_authorisations/PA123
Content-Type: application/json
{
  "payer_authorisations": {
    "id": "PA123",
    "created_at": "2020-04-15T12:00:00Z",
    "status": "created",
    "customer": {
      "email": "mail@example.com",
      "given_name": "Name",
      "family_name": "Surname",
      "metadata": {
        "salesforce_id": "EFGH5678"
      }
    }
  }
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

},
"bank_account": {
  "account_holder_name": "Name Surname",
  "account_numberEnding": "11",
  "bank_name": "BARCLAYS BANK",
  "metadata": {}
},
"mandate": {
  "reference": "XYZ789",
  "metadata": {}
},
"links": {},
"incomplete_fields": [
  {
    "field": "payer_authorisations",
    "message": "is required",
    "request_pointer": "/payer_authorisations/customer/address_line1"
  },
  {
    "field": "payer_authorisations",
    "message": "is required",
    "request_pointer": "/payer_authorisations/customer/city"
  },
  {
    "field": "payer_authorisations",
    "message": "is required",
    "request_pointer": "/payer_authorisations/customer/postal_code"
  }
]
}
}

```

Example response with all properties

```

HTTP/1.1 201 Created
Content-Type: application/json
{
  "payer_authorisations": {
    "id": "PA123",
    "created_at": "2020-04-15T15:00:00.000Z",
    "status": "created",
    "customer": {
      "email": "mail@example.com",
      "given_name": "Name",
      "family_name": "Surname",
      "metadata": {
        "salesforce_id": "EFGH5678"
      }
    },
    "bank_account": {
      "account_holder_name": "Name Surname",
      "account_numberEnding": "11",
      "bank_name": "BARCLAYS BANK",
      "metadata": {}
    },
    "mandate": {
      "reference": "XYZ789",
      "metadata": {}
    },
    "links": {},
    "incomplete_fields": []
  }
}

```

Example response with errors

```

HTTP/1.1 422 Unprocessable entity
Content-Type: application/json
{
  "error": {
    "message": "Validation failed",
    "errors": [
      {
        "field": "bank_account",
        "message": "is the wrong length (should be 8 characters)",
        "request_pointer": "/payer_authorisations/bank_account/account_number"
      },
      {
        "field": "bank_account",
        "message": "is required",
        "request_pointer": "/payer_authorisations/bank_account/branch_code"
      },
      {
        "field": "bank_account",
        "message": "is required",
        "request_pointer": "/payer_authorisations/bank_account/branch_code"
      }
    ],
    "documentation_url": "https://developer.gocardless.com/using-the-api/error-handling/#unprocessable-entity",
    "type": "validation_failed",
    "request_id": "5a83ff47-422e-4f3d-8a2c-1a2a2a2a2a2a",
    "code": 422
  }
}
PHP
$client = new \GoCardlessPro\Client();
$access_token => 'your_access_token';
$environment => \GoCardlessPro\Environment::TEST;
]);
$client->payerAuthorisations()
  >> $params => [
    "customer" => [
      "email" => "mail@example.com",
    ]
  ];

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "given_name" => "Name",
    "family_name" => "Surname",
    "metadata" => [
      "salesforce_id" => "EFGH5678"
    ],
  ],
  "bank_account" => [
    "account_holder_name" => "Name Surname",
    "branch_code" => "200000",
    "account_number" => "55779911",
    "metadata" => []
  ],
  "mandate" => [
    "reference" => "XYZ789",
    "metadata" => []
  ]
]
]);

```

Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payer_authorisations.create(params={
  "customer": {
    "email": "mail@example.com",
    "given_name": "Name",
    "family_name": "Surname",
    "metadata": {}
  },
  "bank_account": {
    "account_holder_name": "Name Surname",
    "branch_code": "200000",
    "account_number": "55779911",
    "metadata": {}
  },
  "mandate": {
    "reference": "XYZ789",
    "metadata": {}
  }
})

```

Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payer_authorisations.create(
  params: {
    "customer": {
      "email": "mail@example.com",
      "given_name": "Name",
      "family_name": "Surname",
      "metadata": {
        "salesforce_id": "EFGH5678"
      }
    },
    "bank_account": {
      "account_holder_name": "Name Surname",
      "branch_code": "200000",
      "account_number": "55779911",
      "metadata": {}
    },
    "mandate": {
      "reference": "XYZ789",
      "metadata": {}
    }
  }
)

```

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

PayerAuthorisation payerAuthorisation = client.payerAuthorisations().create()
  .withCustomerEmail("mail@example.com")
  .withCustomerGivenName("Name")
  .withCustomerFamilyName("Surname")
  .withCustomerMetadata()
  .withBankAccountAccountHolderName("Name Surname")
  .withBankAccountBranchCode("200000")
  .withBankAccountAccountNumber()
  .withBankAccountMetadata()
  .withMandateReference("XYZ789")
  .withMandateMetadata()
  .execute();

```

JavaScript

```

const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const payerAuthorisation = await client.payerAuthorisations().create({
  "customer": {
    "email": "mail@example.com",
    "given_name": "Name",
    "family_name": "Surname",
    "metadata": {}
  },
  "bank_account": {
    "account_holder_name": "Name Surname",
    "branch_code": "200000",
    "account_number": "55779911",
    "metadata": {}
  },
  "mandate": {
    "reference": "XYZ789",
    "metadata": {}
  }
});

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "metadata": {
            "salesforce_id": "EFGH5678"
        }
    },
    "bank_account": {
        "account_holder_name": "Name Surname",
        "branch_code": "200000",
        "account_number": "55779911",
        "metadata": {}
    },
    "mandate": {
        "reference": "XYZ789",
        "metadata": {}
    }
}
});
.NET
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customer = new GoCardless.Services.PayerAuthorisationCreateRequest.PayerAuthorisationCustomer()
{
    "email" = "mail@example.com",
    "given_name" = "Name",
    "family_name" = "Surname",
    "metadata" = {}
};
var bankAccount = new GoCardless.Services.PayerAuthorisationCreateRequest.PayerAuthorisationBankAccount()
{
    "account_holder_name" = "Name Surname",
    "branch_code" = "200000",
    "account_number" = "55779911",
    "metadata" = {}
};
var mandate = new GoCardless.Services.PayerAuthorisationCreateRequest.PayerAuthorisationMandate()
{
    "reference" = "XYZ789",
    "metadata" = {}
};
var PayerAuthorisationRequest = new GoCardless.Services.PayerAuthorisationCreateRequest()
{
    Customer = customer,
    BankAccount = bankAccount,
    Mandate = mandate
};
var PayerAuthorisationResponse = await gocardless.PayerAuthorisations.CreateAsync(PayerAuthorisationRequest);
GoCardless.Resources.PayerAuthorisation payerAuthorisation = PayerAuthorisationResponse.PayerAuthorisation;

```

**Go**

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    payerAuthorisationCreateParams := gocardless.PayerAuthorisationCreateParams{
        Customer: gocardless.PayerAuthorisationCreateParamsCustomer{
            Email:      "mail@example.com",
            GivenName:  "Name",
            FamilyName: "Surname",
            Metadata:   map[string]interface{}{"salesforce_id": "EFGH5678"},
        },
        BankAccount: gocardless.PayerAuthorisationCreateParamsBankAccount{
            AccountHolderName: "Name Surname",
            BranchCode:        "200000",
            AccountNumber:     "55779911",
        },
        Mandate: gocardless.PayerAuthorisationCreateParamsMandate{
            Reference: "XYZ789",
        },
    }

    payerAuthorisation, err := client.PayerAuthorisations.Create(context.Background(), payerAuthorisationCreateParams)
}

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

ly fields present in the request  
you do not need to save any  
at in order to update the metadata

### Update a Payer Authorisation

Updates a Payer Authorisation. **U** will be modified. An empty array **payer data** on your servers or the **l** attribute values it must be sent co

Relative endpoint: PUT /payer\_au

#### Parameters

bank\_account[account\_holder\_name]

 Consultez notre politique de confidentialité

Name of the account holder, as known by the bank. This field will be transliterated, upcased and truncated to 18 characters. This field is required unless the request includes a [customer bank account token](#).

`bank_account[account_number]`

Bank account number - see [local details](#) for more information. Alternatively you can provide an `iban`.

`bank_account[account_number_ending]`

The last few digits of the account number. Currently 4 digits for NZD bank accounts and 2 digits for other currencies.

`bank_account[account_number_suffix]`

Account number suffix (only for bank accounts denominated in NZD) - see [local details](#) for more information.

`bank_account[account_type]`

Bank account type. Required for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.

`bank_account[bank_code]`

Bank code - see [local details](#) for more information. Alternatively you can provide an `iban`.

`bank_account[branch_code]`

Branch code - see [local details](#) for more information. Alternatively you can provide an `iban`.

`bank_account[country_code]`

[ISO 3166-1 alpha-2 code](#). Defaults to the country code of the `iban` if supplied, otherwise is required.

`bank_account[currency]`

[ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

`bank_account[iban]`

International Bank Account Number. Alternatively you can provide [local details](#). IBANs are not accepted for Swedish bank accounts denominated in SEK - you must supply [local details](#).

`bank_account[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer[address_line1]`

The first line of the customer’s address.

`customer[address_line2]`

The second line of the customer’s address.

`customer[address_line3]`

The third line of the customer’s address.

`customer[city]`

The city of the customer’s address.

`customer[company_name]`

Customer’s company name. Required unless a `given_name` and `family_name` are provided. For Canadian customers, the use of a `company_name` value will mean that any mandate created from this customer will be considered to be a “Business PAD” (otherwise, any mandate will be considered to be a “Personal PAD”).

`customer[country_code]`

[ISO 3166-1 alpha-2 code](#).

`customer[danish_identity_number]`

For Danish customers only. The civic/company number (CPR or CVR) of the customer. Must be supplied if the customer’s bank account is denominated in Danish krone (DKK).

`customer[email]`

Customer’s email address. Required in most cases, as this allows GoCardless to send notifications to this customer.

`customer[family_name]`

Customer’s surname. Required unless a `company_name` is provided.

`customer[given_name]`

Customer’s first name. Required unless a `company_name` is provided.

`customer[locale]`

An [IETF Language Tag](#), used for both language and regional variations of our product.

`customer[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`customer[postal_code]`

The customer’s postal code.

`customer[region]`

The customer’s address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

`customer[swedish_identity_number]`

For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Must be supplied if the customer’s bank account is denominated in Swedish krona (SEK). This field cannot be changed once it has been set.

`mandate[metadata]`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`mandate[payer_ip_address]`

For ACH customers only. Required for ACH customers. A string containing the IP address of the payer to whom the mandate belongs (i.e. as a result of their completion of a mandate setup flow in their browser).

Not required for creating offline mandates where `authorisation_source` is set to telephone or paper.

`mandate[reference]`

Unique reference. Different schemes have different length and [character set](#) requirements. GoCardless will generate a unique reference satisfying the different scheme requirements if this field is left blank.

`mandate[scheme]`

A bank payment scheme. Currently “ach”, “autogiro”, “bacs”, “becs”, “becs\_nz”, “betalingsservice”, “faster\_payments”, “pad”, “pay\_to” and “sepa\_core” are supported.



HTTP PHP Python Ruby Java Java

HTTP

```
PUT https://api.gocardless.com/v1/mandates
Content-Type: application/json
{
```

```
  "payer_authorisations": {
    "customer": {
      "email": "mail@example.com",
      "given_name": "Name",
      "family_name": "Surname",
      "metadata": {
        "salesforce_id": "EFGH"
      }
    },
    "bank_account": {
      "account_holder_name": "Name Surname",
      "branch_code": "200000",
      "country_code": "GB"
    }
  }
}
```

## Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

```

        "account_number": "55779911"
    },
    "mandate": {
        "reference": "XYZ789",
    }
}
}

```

Example response with incomplete fields

```

HTTP/1.1 200 OK
Content-Type: application/json
{
    "payer_authorisations": {
        "id": "PA123",
        "created_at": "2020-04-15T15:00:00.000Z",
        "status": "created",
        "customer": {
            "email": "mail@example.com",
            "given_name": "Name",
            "family_name": "Surname",
            "metadata": {
                "salesforce_id": "EFGH5678"
            }
        },
        "bank_account": {
            "account_holder_name": "Name Surname",
            "account_numberEnding": "11",
            "bank_name": "BARCLAYS BANK",
            "metadata": {}
        },
        "mandate": {
            "reference": "XYZ789",
            "metadata": {}
        }
    },
    "links": {},
    "incomplete_fields": [
        {
            "field": "payer_authorisations",
            "message": "is required",
            "request_pointer": "/payer_authorisations/customer/address_line1"
        },
        {
            "field": "payer_authorisations",
            "message": "is required",
            "request_pointer": "/payer_authorisations/customer/city"
        },
        {
            "field": "payer_authorisations",
            "message": "is required",
            "request_pointer": "/payer_authorisations/customer/postal_code"
        }
    ]
}
}

```

Example response where Payer Authorisation is updated after submitting

```

HTTP/1.1 422 Bad Request
Content-Type: application/json
{
    "error": {
        "message": "The Payer Authorisation cannot be updated after submitted",
        "errors": [
            {
                "reason": "payer_authorisation_cannot_update_after_submit",
                "message": "The Payer Authorisation cannot be updated after submitting"
            }
        ],
        "documentation_url": "https://developer.gocardless.com/api-reference#payer_authorisation_cannot_update_after_submit",
        "type": "invalid_state",
        "request_id": "...",
        "code": 422
    }
}

```

Example response with errors

```

HTTP/1.1 422 Unprocessable entity
Content-Type: application/json
{

```

```

    "error": {
        "message": "Validation failed",
        "errors": [
            {
                "field": "customer",
                "message": "is not valid",
                "request_pointer": "/customer"
            }
        ],
        "documentation_url": "https://developer.gocardless.com/api-reference#validation_failed",
        "type": "validation_failed",
        "request_id": "2f0db86c-1234-4567-89ab-cdef01234567",
        "code": 422
    }
}
PHP
$client = new \GoCardlessPro\CCMClient();
$access_token' => 'your_access_token',
'environment'  => \GoCardlessPro\Environment::Production
]);

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
$client->payerAuthorisations()->update("PA123", [
    "params" => ["customer" => [
        "email" => "mail@example.com",
        "given_name" => "Name",
        "family_name" => "Surname",
        "metadata" => {
            "salesforce_id" => "EFGH5678"
        }
    ],
    "bank_account" => [
        "account_holder_name" => "Name Surname",
        "branch_code" => "200000",
        "account_number" => "55779911",
    ],
    "mandate" => [
        "reference" => "XYZ789",
    ]
])
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payer_authorisations.update("PA123", params={
    "customer": {
        "email": "mail@example.com",
        "given_name": "Name",
        "family_name": "Surname",
        "metadata": {
            "salesforce_id": "EFGH5678"
        }
    },
    "bank_account": {
        "account_holder_name": "Name Surname",
        "branch_code": "200000",
        "account_number": "55779911",
    },
    "mandate": {
        "reference": "XYZ789",
    }
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payer_authorisations.update(
  "PA123",
  params: {
    "customer": {
      "email": "mail@example.com",
      "given_name": "Name",
      "family_name": "Surname",
      "metadata": {
        "salesforce_id": "EFGH5678"
      }
    },
    "bank_account": {
      "account_holder_name": "Name Surname",
      "branch_code": "200000",
      "account_number": "55779911",
    },
    "mandate": {
      "reference": "XYZ789",
    }
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

PayerAuthorisation payerAuthorisation = client.payerAuthorisations().update("PA123")
    .withCustomerEmail("mail@example.com")
    .withCustomerGivenName("Name")
    .withCustomerFamilyName("Surname")
    .withCustomerMetadata("salesforce_id", "EFGH5678")
    .withBankAccountAccountHolderName("Name Surname")
    .withBankAccountBranchCode("200000")
    .withBankAccountAccountNumber("55779911")
    .withMandateReference("XYZ789")
    .execute();
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token_here');

const payerAuthorisation = await client.payerAuthorisations().update("PA123", {
    "customer": {
        "email": "mail@example.com",
        "given_name": "Name",
        "family_name": "Surname",
        "metadata": {
            "salesforce_id": "EFGH5678"
        }
    },
    "bank_account": {
        "account_holder_name": "Name Surname",
        "branch_code": "200000",
        "account_number": "55779911"
    },
    "mandate": {
        "reference": "XYZ789"
    }
});
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"family_name": "Surname",
"metadata": {
    "salesforce_id": "EFGH5678"
}
},
"bank_account": {
    "account_holder_name": "Name Surname",
    "branch_code": "200000",
    "account_number": "55779911",
    "metadata": {}
},
"mandate": {
    "reference": "XYZ789",
    "metadata": {}
}
});
.NET
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var customer = new GoCardless.Services.PayerAuthorisationUpdateRequest.PayerAuthorisationCustomer()
{
    "email" = "mail@example.com",
    "given_name" = "Name",
    "family_name" = "Surname",
    "metadata" = {
        "salesforce_id" = "EFGH5678"
    }
};
var bankAccount = new GoCardless.Services.PayerAuthorisationUpdateRequest.PayerAuthorisationBankAccount()
{
    "account_holder_name" = "Name Surname",
    "branch_code" = "200000",
    "account_number" = "55779911",
};
var mandate = new GoCardless.Services.PayerAuthorisationUpdateRequest.PayerAuthorisationMandate()
{
    "reference" = "XYZ789",
};
var PayerAuthorisationRequest = new GoCardless.Services.PayerAuthorisationUpdateRequest()
{
    Customer = customer,
    BankAccount = bankAccount,
    Mandate = mandate
};
var PayerAuthorisationResponse = await gocardless.PayerAuthorisations.UpdateAsync("PA123", PayerAuthorisationRequest);

Go
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

payerAuthorisationUpdateParams := gocardless.PayerAuthorisationUpdateParams{
    Customer: gocardless.PayerAuthorisationUpdateParamsCustomer{
        Email:      "mail@example.com",
        GivenName:  "Name",
        FamilyName: "Surname",
        Metadata:   map[string]interface{}{"salesforce_id": "EFGH5678"},
    },
    BankAccount: gocardless.PayerAuthorisationUpdateParamsBankAccount{
        AccountHolderName: "Name Surname",
        BranchCode:        "200000",
        AccountNumber:     "55779911",
    },
    Mandate: gocardless.PayerAuthorisationUpdateParamsMandate{
        Reference: "XYZ789",
    },
}

payerAuthorisation, err := client.PayerAuthorisations.Update(context_, payerAuthorisationUpdateParams)

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

source is valid and a 422 error

## **Submit a Payer Authorisation**

Submits all the data previously published response in case of validation error

Relative endpoint: POST /payer\_a



HTTP PHP Python Ruby Java Java  
HTTP



[Consultez notre politique de confidentialité](#)

```
POST https://api.gocardless.com/payer_authorisations/PA123/actions/submit HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "payer_authorisations": {
    "id": "PA123",
    "created_at": "2020-04-15T15:00:00.000Z",
    "status": "submitted",
    "customer": {
      "email": "mail@example.com",
      "given_name": "Name",
      "family_name": "Surname",
      "metadata": {
        "salesforce_id": "EFGH5678"
      }
    },
    "bank_account": {
      "account_holder_name": "Name Surname",
      "account_numberEnding": "11",
      "bank_name": "BARCLAYS BANK",
      "metadata": {}
    },
    "mandate": {
      "reference": "XYZ789",
      "metadata": {}
    },
    "links": {},
    "incomplete_fields": []
  }
}
```

or

Example response with errors

```
HTTP/1.1 422 Unprocessable entity
Content-Type: application/json
{
  "error": {
    "message": "Validation failed",
    "errors": [
      {
        "field": "customer",
        "message": "is not valid",
        "request_pointer": "/payer_authorisations/customer/email"
      }
    ],
    "documentation_url": "https://developer.gocardless.com/api-reference#validation_failed",
    "type": "validation_failed",
    "request_id": "2f0db86c-251c-432a-a7af-b77f1a426a37",
    "code": 422
  }
}
```

or

Example response if the resource is already submitted

```
HTTP/1.1 304 Not modified
{}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->payerAuthorisations()->submit("PAU123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payer_authorisations.submit("PAU123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payer_authorisations.submit("PAU123")
```

Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

PayerAuthorisation payerAuthorisation = client
  .payerAuthorisations()
  .submit("PAU123");
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token_here', constants.environments.sandbox);
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

```

const payerAuthorisation = await client.payerAuthorisations.submit("PAU123");
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var payerAuthorisationResponse = await gocardless.payerAuthorisations.SubmitAsync("PAU123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    payerAuthorisation, err := client.PayerAuthorisations.Submit(context, "PAU123")
}

```

## Confirm a Payer Authorisation

Confirms the Payer Authorisation, indicating that the resources are ready to be created. A Payer Authorisation cannot be confirmed if it hasn't been submitted yet.

The main use of the confirm endpoint is to enable integrators to acknowledge the end of the setup process. They might want to make the payers go through some other steps after they go through our flow or make them go through the necessary verification mechanism (upcoming feature).

Relative endpoint: POST /payer\_authorisations/PA123/actions/confirm



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/payer_authorisations/PA123/actions/confirm HTTP/1.1
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
    "payer_authorisations": {
        "id": "PA123",
        "created_at": "2020-04-15T15:00:00.000Z",
        "status": "confirmed",
        "customer": {
            "email": "mail@example.com",
            "given_name": "Name",
            "family_name": "Surname",
            "metadata": {
                "salesforce_id": "EFGH5678"
            }
        },
        "bank_account": {
            "account_holder_name": "Name Surname",
            "account_numberEnding": "11",
            "bank_name": "BARCLAYS BANK",
            "metadata": {}
        },
        "mandate": {
            "reference": "XYZ789",
            "metadata": {}
        },
        "links": {},
        "incomplete_fields": []
    }
}
```

or

```
HTTP/1.1 422 Bad Request
Content-Type: application/json
```

```
{
    "error": {
        "message": "The Payer Authorisation could not be confirmed because it has not been submitted yet.",
        "errors": [
            {
                "reason": "payer_authorisation_not_submitted",
                "message": "The Payer Authorisation has not been submitted yet."
            }
        ],
        "documentation_url": "https://gocardless.com/docs/api/error-handling#422-bad-request",
        "type": "invalid_state",
        "request_id": "...",
        "code": 422
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->payerAuthorisations()->confirm("PAU123");
Python
```

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payer_authorisations.confirm("PAU123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payer_authorisations.confirm("PAU123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

PayerAuthorisation payerAuthorisation = client.payerAuthorisations.confirm("PAU123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payerAuthorisation = await client.payerAuthorisations.confirm("PAU123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var payerAuthorisationResponse = await gocardless.payerAuthorisations.ConfirmAsync("PAU123");
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    payerAuthorisation, err := client.PayerAuthorisations.Confirm(context, "PAU123")
}
```

## Payer Themes

Custom colour themes for payment pages and customer notifications.

Properties

**id**

Unique identifier, beginning with “PTH”.

### Create a payer theme associated with a creditor

Creates a new payer theme associ

Relative endpoint: POST /brandin

**Restricted:** This endpoint is restri

Parameters

**button\_background\_colour**  
Colour for buttons backgrou

**content\_box\_border\_colour**  
Colour for content box bor

**header\_background\_colour**  
Colour for header backgrou

**link\_text\_colour**

e linked to the creditor.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Colour for text links (hexcode)

links[**creditor**]  
 ID of the creditor the payer theme belongs to  
 ○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
 HTTP

POST [https://api.gocardless.com/branding/payer\\_themes](https://api.gocardless.com/branding/payer_themes) HTTP/1.1

Content-Type: application/json

```
{
  "payer_themes": {
    "header_background_colour": "#BD10E0",
    "link_text_colour": "#7ED321",
    "button_background_colour": "#128DAA",
    "content_box_border_colour": "#BD10E0",
    "links": {
      "creditor": "CR123"
    }
  }
}
```

HTTP/1.1 201 OK

Content-Type: application/json

```
{
  "payer_themes": {
    "id": "PTH123"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->payerThemes()->createForCreditor([
  'params' => [
    "header_background_colour" => "#BD10E0",
    "link_text_colour" => "#7ED321",
    "button_background_colour" => "#128DAA",
    "content_box_border_colour" => "#BD10E0",
    "links" => [
      "creditor" => "CR123"
    ]
  ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payer_themes.create_for_creditor(params={
  "header_background_colour": "#BD10E0",
  "link_text_colour": "#7ED321",
  "button_background_colour": "#128DAA",
  "content_box_border_colour": "#BD10E0",
  "links": {
    "creditor": "CR123"
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payer_themes.create_for_creditor(
  params: {
    "header_background_colour": "#BD10E0",
    "link_text_colour": "#7ED321",
    "button_background_colour": "#128DAA",
    "content_box_border_colour": "#BD10E0",
    "links": {
      "creditor": "CR123"
    }
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

PayerTheme payerTheme = client
  .withButtonBackgroundColour(
  .withContentBoxBorderColour(
  .withHeaderBackgroundColour(
  .withLinkTextColour("#7ED321")
  .withLinksCreditor("CR123")
  .execute();
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

JavaScript

```
const constants = require('gocardless-nodejs');
```

 Consultez notre politique de confidentialité

```

const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payerThemes = await client.payerThemes.createForCreditor({
  header_background_colour: "#BD10E0",
  link_text_colour: "#7ED321",
  button_background_colour: "#128DAA",
  content_box_border_colour: "#BD10E0",
  links: {
    creditor: "CR123"
  }
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var payerThemeCreateForCreditorRequest = new GoCardless.Services.PayerThemeCreateForCreditorRequest()
{
  HeaderBackgroundColour = "#BD10E0",
  LinkTextColour = "#7ED321",
  ButtonBackgroundColour = "#128DAA",
  ContentBoxBorderColour = "#BD10E0",
  Links = new GoCardless.Services.PayerThemeCreateForCreditorRequest.PayerThemeLinks()
  {
    Creditor = "CR0123"
  }
};

var payerThemeCreateForCreditorResponse = await gocardless.PayerThemes.CreateForCreditorAsync(payerThemeCreateForCreditorRequest);
GoCardless.Resources.PayerTheme creditorPayerTheme = payerThemeCreateForCreditorResponse.PayerThemeResponse;

Go

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  payerThemeCreateForCreditorParams := gocardless.PayerThemeCreateForCreditorParams{
    ButtonBackgroundColour: "#128DAA",
    ContentBoxBorderColour: "#BD10E0",
    HeaderBackgroundColour: "#BD10E0",
    LinkTextColour: "#7ED321",
    Links: gocardless.PayerThemeCreateForCreditorParamsLinks{
      Creditor: "CR123"
    }
  }

  payerThemes, err := client.PayerThemes.CreateForCreditor(context, payerThemeCreateForCreditorParams)
}

```

## Payments

Payment objects represent payments from a [customer](#) to a [creditor](#), taken against a Direct Debit [mandate](#).

GoCardless will notify you via a [webhook](#) whenever the state of a payment changes.

### Properties

<b>id</b>	Unique identifier, beginning with “PM”.
<b>amount</b>	Amount, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
<b>amount_refunded</b>	Amount <a href="#">refunded</a> , in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
<b>charge_date</b>	A future date on which the payment should be collected. If not specified, the payment will be collected as soon as possible. If the value is before the <a href="#">mandate</a> 's <a href="#">next_possible_charge_date</a> creation will fail. If the value is not a working day it will be rolled forwards to the next available one.
<b>created_at</b>	Fixed <a href="#">timestamp</a> , recording
<b>currency</b>	<a href="#">ISO 4217</a> currency code. Can send its own notifications (e.g. faster_ach)
<b>description</b>	A human-readable description.
<b>faster_ach</b>	This field indicates whether
	It is only present in the API
<b>fx[estimated_exchange_rate]</b>	

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Si votre organisation a des exigences particulières en matière de cookies, veuillez consulter la page de confidentialité pour plus d'informations.



Consultez notre politique de confidentialité

Estimated rate that will be used in the foreign exchange of the `amount` into the `fx_currency`. This will vary based on the prevailing market rate until the moment that it is paid out. Present only before a resource is paid out. Has up to 10 decimal places.

**fx[exchange\_rate]**

Rate used in the foreign exchange of the `amount` into the `fx_currency`. Present only after a resource is paid out. Has up to 10 decimal places.

**fx[fx\_amount]**

Amount that was paid out in the `fx_currency` after foreign exchange. Present only after the resource has been paid out.

**fx[fx\_currency]**

[ISO 4217](#) code for the currency in which amounts will be paid out (after foreign exchange). Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported. Present only if payouts will be (or were) made via foreign exchange.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**reference**

An optional reference that will appear on your customer’s bank statement. The character limit for this reference is dependent on the scheme.

**ACH** - 10 characters

**Autogiro** - 11 characters

**Bacs** - 10 characters

**BECS** - 30 characters

**BECS NZ** - 12 characters

**Betalingsservice** - 30 characters

**Faster Payments** - 18 characters

**PAD** - scheme doesn’t offer references

**PayTo** - 18 characters

**SEPA** - 140 characters

Note that this reference must be unique (for each merchant) for the BECS scheme as it is a scheme requirement. **Restricted:** You can only specify a payment reference for Bacs payments (that is, when collecting from the UK) if you’re on the [GoCardless Plus, Pro or Enterprise packages](#). **Restricted:** You can not specify a payment reference for Faster Payments.

**retry\_if\_possible**

On failure, automatically retry the payment using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

**status**

One of:

- `pending_customer_approval`: we’re waiting for the customer to approve this payment
- `pending_submission`: the payment has been created, but not yet submitted to the banks
- `submitted`: the payment has been submitted to the banks
- `confirmed`: the payment has been confirmed as collected
- `paid_out`: the payment has been included in a [payout](#)
- `cancelled`: the payment has been cancelled
- `customer_approval_denied`: the customer has denied approval for the payment. You should contact the customer directly
- `failed`: the payment failed to be processed. Note that payments can fail after being confirmed if the failure message is sent late by the banks.
- `charged_back`: the payment has been charged back

**links[creditor]**

ID of [creditor](#) to which the collected payment will be sent.

**links[instalment\_schedule]**

ID of [instalment\\_schedule](#) from which this payment was created.

**Note:** this property will only be present if this payment is part of an instalment schedule.

**links[mandate]**

ID of the [mandate](#) against which this payment should be collected.

**links[payout]**

ID of [payout](#) which contains the funds from this payment.

**Note:** this property will not be present until the payment has been successfully collected.

**links[subscription]**

ID of [subscription](#) from which this payment was created.

**Note:** this property will only be present if this payment is part of a subscription.

## Create a payment

Creates a new payment object.

This fails with a `mandate_is_inactive` error if the linked [mandate](#) is cancelled or has failed. Payments can be created against mandates with status of: `pending_customer_approval`, `pending_submission`, `submitted`, and `active`.

Relative endpoint: POST /payments

**Warning:** in Bacs, you have the option to collect payments in multiple currencies (which can only be enabled by configuration). The total amount of all references may be up to 140 characters.

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

custom mandate references  
can't exceed 18 characters. Payment

**Parameters****amount**

*required* Amount, in the lowest

**currency**

*required* [ISO 4217](#) currency

**app\_fee**

The amount to be deducted

**charge\_date**

A future date on which the payment

`next_possible_charge_date` creation will fail. If the value is not a working day it will be rolled forwards to the next available one.

is created.

in GBP, cents in EUR).

the value is before the [mandate](#)'s

**description**

A human-readable description of the payment. This will be included in the notification email GoCardless sends to your customer if your organisation does not send its own notifications (see [compliance requirements](#)).

**faster\_ach**

Set this to true or false in the request to create an ACH payment to explicitly choose whether the payment should be processed through Faster ACH or standard ACH, rather than relying on the presence or absence of the charge date to indicate that.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**reference**

An optional reference that will appear on your customer's bank statement. The character limit for this reference is dependent on the scheme.

**ACH** - 10 characters

**Autogiro** - 11 characters

**Bacs** - 10 characters

**BECS** - 30 characters

**BECS NZ** - 12 characters

**Betalingservice** - 30 characters

**Faster Payments** - 18 characters

**PAD** - scheme doesn't offer references

**PayTo** - 18 characters

**SEPA** - 140 characters

Note that this reference must be unique (for each merchant) for the BECS scheme as it is a scheme requirement. **Restricted:** You can only specify a payment reference for Bacs payments (that is, when collecting from the UK) if you're on the [GoCardless Plus, Pro or Enterprise packages](#). **Restricted:** You can not specify a payment reference for Faster Payments.

**retry\_if\_possible**

On failure, automatically retry the payment using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

**links[mandate]**

required ID of the [mandate](#) against which this payment should be collected.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/payments HTTP/1.1
```

Content-Type: application/json

```
{
  "payments": {
    "amount": 100,
    "currency": "GBP",
    "charge_date": "2014-05-19",
    "reference": "WINEBOX001",
    "metadata": {
      "order_dispatch_date": "2014-05-22"
    },
    "links": {
      "mandate": "MD123"
    }
  }
}
```

HTTP/1.1 201 Created

Location: /payments/PM123

Content-Type: application/json

```
{
  "payments": {
    "id": "PM123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "charge_date": "2014-05-21",
    "amount": 100,
    "description": null,
    "currency": "GBP",
    "status": "pending_submission",
    "reference": "WINEBOX001",
    "metadata": {
      "order_dispatch_date": "2014-05-22"
    },
    "amount_refunded": 0,
    "fx": {
      "fx_currency": "EUR",
      "fx_amount": null,
      "exchange_rate": null,
      "estimated_exchange_rate": "1.1234567890"
    },
    "links": {
      "mandate": "MD123",
      "creditor": "CR123"
    },
    "retry_if_possible": false
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardless\Environment::PRODUCTION;
]);

$client->payments()->create([
  'params' => [
    'amount' => 100,
    'currency' => "GBP",
    'metadata' => [
      'order_dispatch_date' => "2014-05-22"
    ],
    'links' => [
      'mandate' => "MD123"
    ]
  ]
]);
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payments.create(params={
    "amount": 100,
    "currency": "GBP",
    "charge_date": "2014-05-19",
    "reference": "WINEBOX001",
    "metadata": {
        "order_dispatch_date": "2014-05-22"
    },
    "links": {
        "mandate": "MD123"
    }
})
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payments.create(
  params: {
    amount: 1000,
    currency: "GBP",
    description: "Club membership fee",
    links: {
      mandate: "MD123"
    }
  },
  headers: {
    "Idempotency-Key" => "aaf50eb0-8ddb-4900-a97b-40ed794570a1"
  }
)
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

import com.gocardless.services.PaymentService.PaymentCreateRequest.Currency;

Payment payment = client.payments().create()
    .withAmount(100)
    .withCurrency(Currency.GBP)
    .withChargeDate("2014-05-19")
    .withReference("WINEBOX001")
    .withMetadata("order_dispatch_date", "2014-05-22")
    .withLinksMandate("MD123")
    .execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payment = await client.payments.create({
  amount: 100,
  currency: "GBP",
  charge_date: "2014-05-19",
  reference: "WINEBOX001",
  metadata: {
    order_dispatch_date: "2014-05-22"
  },
  links: {
    mandate: "MD123"
  }
});
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var paymentRequest = new GoCardless.Services.PaymentCreateRequest()
{
    Amount = 1000,
    Currency = GoCardless.Services.PaymentCreateRequest.PaymentCurrency.GBP,
    Description = "Membership f
    Links = new GoCardless.Ser
    {
        Mandate = "MD0123"
    }
};

var paymentResponse = await go
GoCardless.Resources.Payment p

Go
package main

import (
    gocardless "github.com/gocardless/gocardless"
)
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    paymentCreateParams := gocardless.PaymentCreateParams{
        Amount:      100,
        Currency:   "GBP",
        ChargeDate: "2014-05-19",
        Reference:  "WINEBOX001",
        Metadata: map[string]interface{}{"order_dispatch_date": "2014-05-22"},
        Links:       gocardless.PaymentCreateParamsLinks{
            Mandate: "MD123",
        },
    }

    payment, err := client.Payments.Create(context, paymentCreateParams)
}

```

## List payments

Returns a [cursor-paginated](#) list of your payments.

Relative endpoint: GET /payments

### Parameters

#### after

Cursor pointing to the start of the desired set.

#### before

Cursor pointing to the end of the desired set.

#### charge\_date[gt]

Limit to records where the payment was or will be collected from the customer's bank account after the specified date.

#### charge\_date[gte]

Limit to records where the payment was or will be collected from the customer's bank account on or after the specified date.

#### charge\_date[lt]

Limit to records where the payment was or will be collected from the customer's bank account before the specified date.

#### charge\_date[lte]

Limit to records where the payment was or will be collected from the customer's bank account on or before the specified date.

#### created\_at[gt]

Limit to records created after the specified date-time.

#### created\_at[gte]

Limit to records created on or after the specified date-time.

#### created\_at[lt]

Limit to records created before the specified date-time.

#### created\_at[lte]

Limit to records created on or before the specified date-time.

#### creditor

ID of a creditor to filter payments by. If you pass this parameter, you cannot also pass customer.

#### currency

[ISO 4217](#) currency code. Currently "AUD", "CAD", "DKK", "EUR", "GBP", "NZD", "SEK" and "USD" are supported.

#### customer

ID of a customer to filter payments by. If you pass this parameter, you cannot also pass creditor.

#### limit

Number of records to return.

#### mandate

Unique identifier, beginning with "MD". Note that this prefix may not apply to mandates created before 2016.

#### sort\_direction

The direction to sort in. One of:

- asc
- desc

#### sort\_field

Field by which to sort records. One of:

- charge\_date
- amount

#### status

One of:

- pending\_customer\_ap
- pending\_submission:
- submitted: the payment
- confirmed: the payment
- paid\_out: the payment

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Consultez notre politique de confidentialité

- **cancelled**: the payment has been cancelled
- **customer\_approval\_denied**: the customer has denied approval for the payment. You should contact the customer directly
- **failed**: the payment failed to be processed. Note that payments can fail after being confirmed if the failure message is sent late by the banks.
- **charged\_back**: the payment has been charged back

**subscription**

Unique identifier, beginning with “SB”.

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET <https://api.gocardless.com/payments> HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "payments": [
    {
      "id": "PM123",
      "created_at": "2014-05-08T17:01:06.000Z",
      "charge_date": "2014-05-15",
      "amount": 100,
      "description": null,
      "currency": "GBP",
      "status": "pending_submission",
      "reference": "WINEBOX001",
      "metadata": {
        "order_dispatch_date": "2014-05-22"
      },
      "amount_refunded": 0,
      "fx": {
        "fx_currency": "EUR",
        "fx_amount": null,
        "exchange_rate": null,
        "estimated_exchange_rate": "1.1234567890"
      },
      "links": {
        "mandate": "MD123",
        "creditor": "CR123"
      },
      "retry_if_possible": false
    }
  ]
}
```

**PHP**

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->payments()->list();

$client->payments()->list([
  'params' => ["customer" => "CU123"]
]);
```

**Python**

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payments.list().records

client.payments.list(params={"customer": "CU123"}).records
```

**Ruby**

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payments.list

@client.payments.list(params: { customer: "CU123" })
```

**Java**

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

for (Payment payment : client.list())
  System.out.println(payment);
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payments = await client.payments.list();

// List all payments past a certain date.
const payments = await client.payments.list({
  created_at: {
    gt: "2020-01-01T17:01:06.000Z"
  }
});
.NET
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var paymentRequest = new GoCardless.Services.PaymentListRequest()
{
  Customer = "CU000123"
};

var paymentListResponse = gocardless.Payments.All(paymentRequest);
foreach (GoCardless.Resources.Payment payment in paymentListResponse)
{
  Console.WriteLine(payment.Id);
}

Go
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  paymentListParams := gocardless.PaymentListParams{
    CreatedAt: &gocardless.PaymentListParamsCreatedAt{
      Gt: "2020-01-01T17:01:06.000Z",
    },
  }

  paymentListResult, err := client.Payments.List(context, paymentListParams)
  for _, payment := paymentListResult.Payments {
    fmt.Println(payment.Id)
  }
}
}

```

## Get a single payment

Retrieves the details of a single existing payment.

Relative endpoint: GET /payments/PM123

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/payments/PM123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "payments": [
    {
      "id": "PM123",
      "created_at": "2014-05-08T17:01:06.000Z",
      "charge_date": "2014-05-15",
      "amount": 100,
      "description": null,
      "currency": "GBP",
      "status": "pending_submission",
      "reference": "WINEBOX001",
      "metadata": {
        "order_dispatch_date": ""
      },
      "amount_refunded": 0,
      "fx": {
        "fx_currency": "EUR",
        "fx_amount": null,
        "exchange_rate": null,
        "estimated_exchange_rate": null
      },
      "links": {
        "mandate": "MD123",
        "creditor": "CR123"
      },
      "retry_if_possible": false
    }
  ]
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité



```

        "metadata": {
          "key": "value"
        }
      }

HTTP/1.1 200 OK
Content-Type: application/json
{
  "payments": {
    "id": "PM123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "charge_date": "2014-05-15",
    "amount": 100,
    "description": null,
    "currency": "GBP",
    "status": "pending_submission",
    "reference": "WINEBOX001",
    "metadata": {
      "key": "value"
    },
    "amount_refunded": 0,
    "fx": {
      "fx_currency": "EUR",
      "fx_amount": null,
      "exchange_rate": null,
      "estimated_exchange_rate": "1.1234567890"
    },
    "links": {
      "mandate": "MD123",
      "creditor": "CR123"
    },
    "retry_if_possible": false
  }
}

```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->payments()->update("PM123", [
  "params" => ["metadata" => ["internal_id" => "XX123"]]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payments.update("PM123", params={
  "metadata": {"key": "value"}
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payments.update(
  "PM123",
  params: {
    metadata: { order_id: "transaction-0HE9WQ0WDE" }
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.payments().update("PM123")
  .withMetadata("order_id", "transaction-0HE9WQ0WDE")
  .execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here').constants.Environments.Sandbox:
```

```
const payment = await client.p
  "PM123",
  {
    metadata: {
      key: "value"
    }
  }
);
```

.NET

```
String accessToken = "your_ac
GoCardlessClient gocardless =
  var paymentRequest = new GoCar
  {
    Metadata = new Dictionary<string, string>()
  }
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

{
    {"invoice_id", "INVOICE338"}
}

};

var paymentResponse = await gocardless.Payments.UpdateAsync("PM0123", paymentRequest);
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    paymentUpdateParams := gocardless.PaymentUpdateParams{
        Metadata: map[string]interface{}{"key": "value"},
    }

    payment, err := client.Payments.Update(context, "PM123", paymentUpdateParams)
}

```

## Cancel a payment

Cancels the payment if it has not already been submitted to the banks. Any metadata supplied to this endpoint will be stored on the payment cancellation event it causes.

This will fail with a `cancellation_failed` error unless the payment's status is `pending_submission`.

Relative endpoint: POST `/payments/PM123/actions/cancel`

Parameters

### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/payments/PM123/actions/cancel HTTP/1.1
Content-Type: application/json
{
    "data": {
        "metadata": {
            "ticket_id": "TK123"
        }
    }
}
```

HTTP/1.1 200 OK
Content-Type: application/json

```
{
    "payments": {
        "id": "PM123",
        "created_at": "2014-05-08T17:01:06.000Z",
        "charge_date": "2014-05-21",
        "amount": 100,
        "description": null,
        "currency": "GBP",
        "status": "cancelled",
        "reference": "WINEBOX001",
        "metadata": {
            "order_dispatch_date": "2014-05-22"
        },
        "amount_refunded": 0,
        "fx": {
            "fx_currency": "EUR",
            "fx_amount": null,
            "exchange_rate": null,
            "estimated_exchange_rate": null
        },
        "links": {
            "mandate": "MD123",
            "creditor": "CR123"
        },
        "retry_if_possible": false
    }
}
```

PHP

```
$client = new \GoCardlessPro\CCP();
'access_token' => 'your_access_token',
'environment'  => \GoCardless\Environment::SANDBOX
]);
```

```
$client->payments()->cancel("PM123");
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payments.cancel("PM123")
```

Ruby

```
@client = GoCardlessPro::Client.new(  
  access_token: "your_access_token",  
  environment: :sandbox  
)
```

@client

```
Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.payments().cancel("PM123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const paymentResponse = await client.payments.cancel("PM123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient goCardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX)
```

```
var paymentResponse = await gocardless.Payments.CancelAsync("PM0123");
```

Go

```
package main
```

```
import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)
func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    paymentCancelParams := gocardless.PaymentCancelParams{}
    payment, err := client.Payments.Cancel(context, "PM123", paymentCancelParams)
}
```

## Retry a payment

Retries a failed payment if the underlying mandate is active. You will receive a `resubmission_requested` webhook, but after that retrying the payment follows the same process as its initial creation, so you will receive a `submitted` webhook, followed by a `confirmed` or `failed` event. Any metadata supplied to this endpoint will be stored against the payment submission event it causes.

This will return a retry, failed error if the payment has not failed

Payments can be retried up to 3 times

Relative endpoint: POST /payments/RM123/actions/potpw

## Parameters

change date

A future date on which the p  
next\_possible\_charge\_date  
metadata  
Key-value store of system

B1 B2 B3 B4 B5 B6 B7 B8

**HTTP**

```
POST https://api.gocardless.com/v1/charges  
Content-Type: application/json  
{  
  "data": {  
    "metadata": {  
      "reason": "Customer requested a refund"  
    }  
  }  
}
```

expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

 Consultez notre politique de cookies

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

}
HTTP/1.1 200 OK
Content-Type: application/json
{
  "payments": {
    "id": "PM123",
    "created_at": "2014-05-08T17:01:06.000Z",
    "charge_date": "2014-05-21",
    "amount": 100,
    "description": null,
    "currency": "GBP",
    "status": "submitted",
    "reference": "WINEBOX001",
    "metadata": {
      "order_dispatch_date": "2014-05-22"
    },
    "amount_refunded": 0,
    "fx": {
      "fx_currency": "EUR",
      "fx_amount": null,
      "exchange_rate": null,
      "estimated_exchange_rate": "1.1234567890"
    },
    "links": {
      "mandate": "MD123",
      "creditor": "CR123"
    },
    "retry_if_possible": false
  }
}

```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->payments()->retry("PM123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payments.retry("PM123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payments.retry("PM123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.payments().retry("PM123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const paymentResponse = await client.payments.retry("PM123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var paymentResponse = await gocardless.Payments.RetryAsync("PM0123");
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless"
)

func main() {
  accessToken := "your_access_token"
  config, err := gocardless.NewConfig()
  if err != nil {
    fmt.Printf("got err in init: %v\n", err)
    return
  }
  client, err := gocardless.NewClient(config)
  if err != nil {
    fmt.Println("error in init: %v", err)
    return
  }

  paymentRetryParams := gocardless.PaymentRetryParams{
    Payment: payment,
    Err:     err,
  }
  payment, err := client.Payments.Retry(context, "PM123", paymentRetryParams)
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

}

## Payouts

Payouts represent transfers from GoCardless to a [creditor](#). Each payout contains the funds collected from one or many [payments](#). All the payments in a payout will have been collected in the same currency. Payouts are created automatically after a payment has been successfully collected.

### Properties

**id**

Unique identifier, beginning with “PO”.

**amount**

Amount in minor unit (e.g. pence in GBP, cents in EUR).

**arrival\_date**

Date the payout is due to arrive in the creditor’s bank account. One of:

- `yyyy-mm-dd`: the payout has been paid and is due to arrive in the creditor’s bank account on this day
- `null`: the payout hasn’t been paid yet

**created\_at**

Fixed [timestamp](#), recording when this resource was created.

**currency**

[ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

**deducted\_fees**

Fees that have already been deducted from the payout amount in minor unit (e.g. pence in GBP, cents in EUR), inclusive of tax if applicable.

For each `late_failure_settled` or `chargeback_settled` action, we refund the transaction fees in a payout. This means that a payout can have a negative `deducted_fees` value.

This field is calculated as (`GoCardless fees + app fees + surcharge fees`) - (`refunded fees`)

If the merchant is invoiced for fees separately from the payout, then `deducted_fees` will be 0.

**fx[estimated\_exchange\_rate]**

Estimated rate that will be used in the foreign exchange of the `amount` into the `fx_currency`. This will vary based on the prevailing market rate until the moment that it is paid out. Present only before a resource is paid out. Has up to 10 decimal places.

**fx[exchange\_rate]**

Rate used in the foreign exchange of the `amount` into the `fx_currency`. Present only after a resource is paid out. Has up to 10 decimal places.

**fx[fx\_amount]**

Amount that was paid out in the `fx_currency` after foreign exchange. Present only after the resource has been paid out.

**fx[fx\_currency]**

[ISO 4217](#) code for the currency in which amounts will be paid out (after foreign exchange). Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported. Present only if payouts will be (or were) made via foreign exchange.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters. *Note:* This should not be used for storing PII data.

**payout\_type**

Whether a payout contains merchant revenue or partner fees.

**reference**

Reference which appears on the creditor’s bank statement.

**status**

One of:

- `pending`: the payout has been created, but not yet sent to your bank or it is in the process of being exchanged through our FX provider.
- `paid`: the payout has been sent to your bank. FX payouts will become `paid` after we emit the `fx_rate_confirmed` webhook.
- `bounced`: the payout bounced when sent, the payout can be retried.

**tax\_currency**

[ISO 4217](#) code for the currency in which tax is paid out to the tax authorities of your tax jurisdiction. Currently “EUR”, “GBP”, for French or British merchants, this will be `null` if tax is not applicable *beta*

**links[creditor]**

ID of [creditor](#) who will receive this payout, i.e. the owner of the `creditor_bank_account`.

**links[creditor\_bank\_account]**

ID of [bank account](#) which this will be sent to.

## List payouts

Returns a [cursor-paginated](#) list of your payouts.

Relative endpoint: GET `/payouts`

### Parameters

**after**

Cursor pointing to the start of

**before**

Cursor pointing to the end of

**created\_at[gt]**

Limit to records created after

**created\_at[gte]**

Limit to records created on

**created\_at[lte]**

Limit to records created before

**created\_at[lte]**

Limit to records created on

**creditor**

Unique identifier, beginning with `www` .

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**creditor\_bank\_account**

Unique identifier, beginning with “BA”.

**currency**

[ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

**limit**

Number of records to return.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters. *Note:* This should not be used for storing PII data.

**layout\_type**

Whether a payout contains merchant revenue or partner fees.

**reference**

Reference which appears on the creditor’s bank statement.

**status**

One of:

- pending: the payout has been created, but not yet sent to your bank or it is in the process of being exchanged through our FX provider.
- paid: the payout has been sent to the your bank. FX payouts will become paid after we emit the `fx_rate_confirmed` webhook.
- bounced: the payout bounced when sent, the payout can be retried.



HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

GET <https://api.gocardless.com/payouts> HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "payouts": [
    {
      "id": "P0123",
      "amount": 1000,
      "arrival_date": "2014-06-27",
      "deducted_fees": 10,
      "currency": "GBP",
      "created_at": "2014-06-20T13:23:34.000Z",
      "payout_type": "merchant",
      "reference": "ref-1",
      "status": "pending",
      "fx": {
        "fx_currency": "EUR",
        "fx_amount": null,
        "exchange_rate": null,
        "estimated_exchange_rate": "1.11667"
      },
      "tax_currency": "GBP",
      "metadata": { "key": "value" },
      "links": {
        "creditor_bank_account": "BA123",
        "creditor": "CR123"
      }
    ],
    "meta": {
      "cursors": {
        "after": null,
        "before": null
      },
      "limit": 50
    }
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->payouts()->list();

$client->payouts()->list([
  'params' => [ "creditor" => "CR123" ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payouts.list().records

client.payouts.list(params={"status": "pending"})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payouts.list

@client.payouts.list(params: {
  status: "pending"
})

@client.payouts.list.records.each do |payout|
```

**Nous utilisons des cookies**

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

for (Payout payout : client.payouts().all().withStatus("pending").execute()) {
    System.out.println(payout.getId());
}

```

JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payouts = await client.payouts.list();

// List all payouts with a given status.
await client.payouts.list({ status: "pending" });

```

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var payoutRequest = new GoCardless.Services.PayoutListRequest()
{
    Currency = GoCardless.Services.PayoutListRequest.PayoutCurrency.EUR,
};

var payoutListResponse = gocardless.Payouts.All(payoutRequest);
foreach (GoCardless.Resources.Payout payout in payoutListResponse)
{
    Console.WriteLine(payout.Amount);
}

```

Go

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    payoutListParams := gocardless.PayoutListParams{}
    payoutListResult, err := client.Payouts.List(context, payoutListParams)
    for _, payout := range payoutListResult.Payouts {
        fmt.Println(payout.Amount)
    }
}

```

## Get a single payout

Retrieves the details of a single payout. For an example of how to reconcile the transactions in a payout, see [this guide](#).

Relative endpoint: GET /payouts/P0123



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/payouts/P0123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "payouts": [
        {
            "id": "P0123",
            "amount": 1000,
            "arrival_date": "2014-06-27T00:00:00Z",
            "deducted_fees": 10,
            "currency": "GBP",
            "created_at": "2014-06-20T14:45:00Z",
            "payout_type": "merchant",
            "reference": "ref-1",
            "status": "pending",
            "fx": {
                "fx_currency": "EUR",
                "fx_amount": null,
                "exchange_rate": null,
                "estimated_exchange_rate": null
            },
            "tax_currency": "GBP",
            "metadata": {
                "key": "value"
            },
            "links": {
                "creditor_bank_account": "CR123"
            }
        }
    ]
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    }
}
}

PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
$client->payouts()->get("P0123");

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payouts.get("P0123")

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payouts.get("P0123")

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.payouts().get("P0123").execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payout = await client.payouts.find("P0123");

```

## .NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var payoutResponse = await gocardless.Payouts.GetAsync("P00123");
GoCardless.Resources.Payout payout = payoutResponse.Payout;

```

## Go

```

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    payout, err := client.Payouts.Get(context, "P0123")
}

```

**Update a payout**

Updates a payout object. This accepts only the metadata parameter.

Relative endpoint: PUT /payouts/P0123

## Parameters

## metadata

Key-value store of custom parameters.

HTTP PHP Python Ruby Java JavaScript

HTTP

PUT <https://api.gocardless.com/payouts/P0123>  
Content-Type: application/json

```
{
    "payouts": [
        "metadata": {
            "key": "value"
        }
    ]
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

rs.

```

    }
}

HTTP/1.1 200 OK
Content-Type: application/json
{
  "payouts": {
    "id": "P0123",
    "amount": 1000,
    "arrival_date": "2014-06-27",
    "deducted_fees": 10,
    "currency": "GBP",
    "created_at": "2014-06-20T13:23:34.000Z",
    "payout_type": "merchant",
    "reference": "ref-1",
    "status": "pending",
    "fx": {
      "fx_currency": "EUR",
      "fx_amount": null,
      "exchange_rate": null,
      "estimated_exchange_rate": "1.11667"
    },
    "tax_currency": "GBP",
    "metadata": {"key": "value"},
    "links": {
      "creditor_bank_account": "BA123",
      "creditor": "CR123"
    }
  }
}

```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->payouts()->update("P0123", [
  "params" => ["metadata" => ["key" => "value"]]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payouts.update("P0123", params={
  "metadata": {"key": "value"}
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payouts.update(
  "P0123",
  params: {
    metadata: { key: "value" }
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.payouts().update("P0123")
  .withMetadata("key", "value")
  .execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const payout = await client.payouts.update(
  "P0123",
  {
    metadata: {
      key: "value"
    }
);

```

.NET

```
String accessToken = "your_ac
GoCardlessClient gocardless =
var payoutRequest = new GoCard
{
  Metadata = new Dictionary<
  {
    {"key", "value"}
  }
};
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var payoutResponse = await gocardless.Payouts.UpdateAsync("P0123", payoutRequest);
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    payoutUpdateParams := gocardless.PayoutUpdateParams{
        Metadata: map[string]interface{}{"key": "value"},
    }

    payout, err := client.Payouts.Update(context, "P0123", payoutUpdateParams)
}

```

## Payout Items

When we collect a payment on your behalf, we add the money you've collected to your GoCardless balance, minus any fees paid. Periodically (usually every working day), we take any positive balance in your GoCardless account, and pay it out to your nominated bank account.

Other actions in your GoCardless account can also affect your balance. For example, if a customer charges back a payment, we'll deduct the payment's amount from your balance, but add any fees you paid for that payment back to your balance.

The Payout Items API allows you to view, on a per-payout basis, the credit and debit items that make up that payout's amount. Payout items can only be retrieved for payouts created in the last 6 months. Requests for older payouts will return an HTTP status `410 Gone`.

### Properties

#### amount

The positive (credit) or negative (debit) value of the item, in fractional currency; the lowest denomination for the currency (e.g. pence in GBP, cents in EUR), to one decimal place. For accuracy, we store some of our fees to greater precision than we can actually pay out (for example, a GoCardless fee we record might come to 0.5 pence, but it is not possible to send a payout via bank transfer including a half penny).

To calculate the final amount of the payout, we sum all of the items and then round to the nearest currency unit.

#### taxes

An array of tax items *beta*

*Note:* VAT applies to transaction and surcharge fees for merchants operating in the UK and France.

Each instance will contain these properties:

- **amount:** The amount of tax applied to a fee in fractional currency; the lowest denomination for the currency (e.g. pence in GBP, cents in EUR), to one decimal place.
- **currency:** [ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.
- **destination\_amount:** The amount of tax to be paid out to the tax authorities in fractional currency; the lowest denomination for the currency (e.g. pence in GBP, cents in EUR), to one decimal place.

When `currency` and `destination_currency` don't match this will be `null` until the `exchange_rate` has been finalised.

- **destination\_currency:** [ISO 4217](#) code for the currency in which tax is paid out to the tax authorities of your tax jurisdiction. Currently “EUR” for French merchants and “GBP” for British merchants.
- **exchange\_rate:** The exchange rate for the tax from the currency into the destination currency.

Present only if the currency and the destination currency don't match and the exchange rate has been finalised.

You can listen for the payout's [tax\\_exchange\\_rates\\_confirmed\\_webhook](#) to know when the exchange rate has been finalised for all fees in the payout.

- **tax\_rate\_id:** The unique identifier created by the jurisdiction, tax type and version

#### type

The type of the credit (positive)

- `payment_paid_out` (credit)
- `payment_failed` (debit)
- `payment_charged_back` (debit)
- `payment_refunded` (debit)
- `refund` (debit): A refund of a payment.
- `refund_funds_returned` (debit): A refund of a payment that has been returned to the merchant.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- `gocardless_fee` (credit/debit): The fees that GoCardless charged for a payment. In the case of a payment failure or chargeback, these will appear as credits. Will include taxes if applicable for merchants.
- `app_fee` (credit/debit): The optional fees that a partner may have taken for a payment. In the case of a payment failure or chargeback, these will appear as credits.
- `revenue_share` (credit/debit): A share of the fees that GoCardless collected which some partner integrations receive when their users take payments. Only shown in partner payouts. In the case of a payment failure or chargeback, these will appear as credits.
- `surcharge_fee` (credit/debit): GoCardless deducted a surcharge fee as the payment failed or was charged back, or refunded a surcharge fee as the bank or customer cancelled the chargeback. Will include taxes if applicable for merchants.

**links[mandate]**

Unique identifier, beginning with “MD”. Note that this prefix may not apply to mandates created before 2016. Present only for the items of type `payment_refunded`, `refund` and `refund_funds_returned`.

**links[payment]**

Unique identifier, beginning with “PM”.

**links[refund]**

Unique identifier, beginning with “RF”. Present only for the items of type `payment_refunded`, `refund` and `refund_funds_returned`.

**Get all payout items in a single payout**

Returns a [cursor-paginated](#) list of items in the payout.

**This endpoint only serves requests for payouts created in the last 6 months. Requests for older payouts will return an HTTP status 410 Gone.**

Relative endpoint: GET /payout\_items

## Parameters

`payout`  
required Unique identifier, beginning with “PO”.

`after`  
Cursor pointing to the start of the desired set.

`before`  
Cursor pointing to the end of the desired set.

`include_2020_tax_cutover`  
Boolean value indicating whether the API should return tax data for the cutover period of April to August 2020. Defaults to false.

`limit`  
Number of records to return.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/payout\_items?payout=P0123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "payout_items": [
    {
      "amount": "4.0",
      "type": "gocardless_fee",
      "taxes": [
        {
          "amount": "1.0",
          "currency": "GBP",
          "destination_amount": "1.1",
          "destination_currency": "EUR",
          "exchange_rate": "1.11205",
          "tax_rate_id": "FR_VAT_1"
        }
      ],
      "links": {
        "payment": "PM456"
      }
    },
    {
      "amount": "1000.0",
      "type": "payment_paid_out",
      "taxes": [],
      "links": {
        "payment": "PM123"
      }
    },
    {
      "amount": "-24.0",
      "type": "gocardless_fee",
      "taxes": [
        {
          "amount": "4.0",
          "currency": "EUR",
          "destination_amount": null,
          "destination_currency": null,
          "exchange_rate": null,
          "tax_rate_id": "FR_VAT"
        }
      ],
      "links": {
        "payment": "PM123"
      }
    },
    {
      "amount": "30.0",
      "type": "payment_paid_out",
      "taxes": [],
      "links": {
        "payment": "PM456"
      }
    }
  ]
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
{
  "amount": "-500.0",
  "type": "payment_refunded",
  "taxes": [],
  "links": {
    "payment": "PM789"
  }
},
{
  "amount": "4.0",
  "type": "gocardless_fee",
  "taxes": [
    {
      "amount": "1.0",
      "currency": "GBP",
      "destination_amount": "1.1",
      "destination_currency": "EUR",
      "exchange_rate": "1.11205",
      "tax_rate_id": "FR_VAT_1"
    }
  ],
  "links": {
    "payment": "PM456"
  }
},
{
  "amount": "-1.0",
  "type": "app_fee",
  "taxes": [],
  "links": {
    "payment": "PM456"
  }
},
],
"meta": {
  "cursors": { "before": null, "after": null },
  "limit": 50
}
}
```

GET [https://api.gocardless.com/payout\\_items?payout=P0456](https://api.gocardless.com/payout_items?payout=P0456) HTTP/1.1

HTTP/1.1 410 Gone  
Content-Type: application/json

```
{
  "error": {
    "documentation_url": "https://developer.gocardless.com/api-reference#payout_items_data_archived",
    "message": "Payout items for payouts created more than 6 months ago are archived. Please contact support if you require access to this data.",
    "type": "invalid_api_usage",
    "errors": [
      {
        "reason": "payout_items_data_archived",
        "message": "Payout items for payouts created more than 6 months ago are archived. Please contact support if you require access to this data."
      }
    ],
    "code": 410,
    "request_id": "deadbeef-0000-4000-0000-444400004444"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->payoutItems()->list([
  'params' => ['payout' => 'P0123']
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.payout_items.list(params={'payout': 'P0123'})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.payout_items.list(params: {payout: "P0123"})
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_acce
GoCardlessClient client = GoCa
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.payoutItems().all().wi
```

JavaScript

```
const constants = require('goc
const gocardless = require('go
const client = gocardless('you

const payoutItems = await clien
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
// List the first three payout items for a payout
const payoutItems = await client.payoutItems.list({ payout: "P0123", limit: 3 });

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var itemsRequest = new GoCardless.Services.PayoutItemsListRequest()
{
    Payout = "P0123"
};

var response = gocardless.PayoutItems.All(payoutRequest);
foreach (GoCardless.Resources.PayoutItem item in response)
{
    Console.WriteLine(item.Amount);
}

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    payoutItemListParams := gocardless.PayoutItemListParams{
        Payout: "P0123",
    }

    payoutItemListResult, err := client.PayoutItems.List(context, payoutItemListParams)
    for _, payoutItem := payoutItemListResult.PayoutItems {
        fmt.Println(payoutItem.Amount)
    }
}
```

## Redirect Flows

**Deprecated:** Redirect Flows are legacy APIs and cannot be used by new integrators. The [Billing Request flow](#) API should be used for your payment flows.

Redirect flows enable you to use GoCardless' [hosted payment pages](#) to set up mandates with your customers. These pages are fully compliant and have been translated into Danish, Dutch, French, German, Italian, Norwegian, Portuguese, Slovak, Spanish and Swedish.

The overall flow is:

1. You [create](#) a redirect flow for your customer, and redirect them to the returned redirect url, e.g. <https://pay.gocardless.com/flow/RE123>.
2. Your customer supplies their name, email, address, and bank account details, and submits the form. This securely stores their details, and redirects them back to your `success_redirect_url` with `redirect_flow_id=RE123` in the querystring.
3. You [complete](#) the redirect flow, which creates a [customer](#), [customer bank account](#), and [mandate](#), and returns the ID of the mandate. You may wish to create a [subscription](#) or [payment](#) at this point.

Once you have [completed](#) the redirect flow via the API, you should display a confirmation page to your customer, confirming that their Direct Debit has been set up. You can build your own page, or redirect to the one we provide in the `confirmation_url` attribute of the redirect flow.

Redirect flows expire 30 minutes after they are first created. You cannot complete an expired redirect flow. For an integrator this is shorter and they will expire after 10 minutes.

### Properties

`id`  
Unique identifier, beginning with “RE”.

`confirmation_url`  
The URL of a confirmation page, which you may optionally redirect the customer to rather than use your own page, that confirms in their chosen language that their Direct Debit has been set up successfully. Only returned once the customer has set up their mandate via the payment pages and the redirect flow has been [completed](#), and only available for 15 minutes from when you complete the redirect flow. The structure of this URL may change at any time, so you should read it directly from the API response.

`created_at`  
Fixed [timestamp](#), recording

`description`  
A description of the item that generates the mandate.

`mandate_reference`  
Mandate reference generated by the API.

`metadata`  
Key-value store of custom data for storing PII data.

`redirect_url`  
The URL of the hosted payment page scheme.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

rs. Note: This should not be used

Consultez notre politique de confidentialité



The Direct Debit scheme of the mandate. If specified, the payment pages will only allow the set-up of a mandate for the specified scheme. It is recommended that you leave this blank so the most appropriate scheme is picked based on the customer's bank account.

**session\_token**

The customer's session ID must be provided when the redirect flow is set up and again when it is completed. This allows integrators to ensure that the user who was originally sent to the GoCardless payment pages is the one who has completed them.

**success\_redirect\_url**

The URL to redirect to upon successful mandate setup. You must use a URL beginning `https` in the live environment.

**links[billing\_request]**

ID of [billing request](#) that a redirect flow can create.

**Note:** The redirect flow will only create a billing request in the event the redirect flow is eligible to send the payer down this new and improved flow

**links[creditor]**

The [creditor](#) for whom the mandate will be created. The name of the creditor will be displayed on the payment page.

**links[customer]**

ID of [customer](#) created by this redirect flow.

**Note:** this property will not be present until the redirect flow has been successfully completed.

**links[customer\_bank\_account]**

ID of [customer bank account](#) created by this redirect flow.

**Note:** this property will not be present until the redirect flow has been successfully completed.

**links[mandate]**

ID of [mandate](#) created by this redirect flow.

**Note:** this property will not be present until the redirect flow has been successfully completed.

## Create a redirect flow

Creates a redirect flow object which can then be used to redirect your customer to the GoCardless hosted payment pages.

Relative endpoint: POST `/redirect_flows`

**Deprecated:** Redirect Flows are legacy APIs and cannot be used by new integrators. The [Billing Request flow](#) API should be used for your payment flows.

**Parameters****session\_token**

**required** The customer's session ID must be provided when the redirect flow is set up and again when it is completed. This allows integrators to ensure that the user who was originally sent to the GoCardless payment pages is the one who has completed them.

**success\_redirect\_url**

**required** The URL to redirect to upon successful mandate setup. You must use a URL beginning `https` in the live environment.

**description**

A description of the item the customer is paying for. This will be shown on the hosted payment pages.

**metadata**

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters. **Note:** This should not be used for storing PII data.

**prefilled\_bank\_account[account\_type]**

Bank account type for USD-denominated bank accounts. Must not be provided for bank accounts in other currencies. See [local details](#) for more information.

**prefilled\_customer[address\_line1]**

The first line of the customer's address.

**prefilled\_customer[address\_line2]**

The second line of the customer's address.

**prefilled\_customer[address\_line3]**

The third line of the customer's address.

**prefilled\_customer[city]**

The city of the customer's address.

**prefilled\_customer[company\_name]**

Customer's company name. Company name should only be provided if `given_name` and `family_name` are null.

**prefilled\_customer[country\_code]**

[ISO 3166-1 alpha-2 code](#).

**prefilled\_customer[danish\_identity\_number]**

For Danish customers only. The civic/company number (CPR or CVR) of the customer.

**prefilled\_customer[email]**

Customer's email address.

**prefilled\_customer[family\_name]**

Customer's surname.

**prefilled\_customer[given\_name]**

Customer's first name.

**prefilled\_customer[language]**

[ISO 639-1](#) code.

**prefilled\_customer[phone\_number]**

For New Zealand customers only.

**prefilled\_customer[postal\_code]**

The customer's postal code.

**prefilled\_customer[region]**

The customer's address region, county or department.

**prefilled\_customer[swedish\_identity\_number]**

For Swedish customers only.

**scheme**

The Direct Debit scheme of the mandate. If specified, the payment pages will only allow the set-up of a mandate for the specified scheme. It is recommended that you leave this blank so the most appropriate scheme is picked based on the customer's bank account.

**links[creditor]**

The [creditor](#) for whom the mandate will be created. The name of the creditor will be displayed on the payment page.



HTTP PHP Python Ruby Java Java

HTTP

```
POST https://api.gocardless.com/mandates
Content-Type: application/json
{
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

he customer.

fied scheme. It is recommended

if your account manages multiple

```

"redirect_flows": {
  "description": "Wine boxes",
  "session_token": "SESS_wSs0uGYMISxzqOBq",
  "success_redirect_url": "https://example.com/pay/confirm",
  "prefilled_customer": {
    "given_name": "Frank",
    "family_name": "Osborne",
    "email": "frank.osborne@acmeplc.com"
  },
  "links": {
    "creditor": "CR123"
  }
}
}

```

```

HTTP/1.1 201 Created
Location: /redirect_flows/RE123
Content-Type: application/json
{
  "redirect_flows": {
    "id": "RE123",
    "description": "Wine boxes",
    "session_token": "SESS_wSs0uGYMISxzqOBq",
    "scheme": null,
    "success_redirect_url": "https://example.com/pay/confirm",
    "redirect_url": "http://pay.gocardless.com/flow/RE123",
    "created_at": "2014-10-22T13:10:06.000Z",
    "links": {
      "creditor": "CR123"
    }
  }
}

```

PHP

```

$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->redirectFlows()->create([
  "params" => ["description" => "Wine boxes",
    "session_token" => "SESS_wSs0uGYMISxzqOBq",
    "success_redirect_url" => "https://example.com/pay/confirm",
    "prefilled_customer" => [
      "given_name" => "Frank",
      "family_name" => "Osborne",
      "email" => "frank.osborne@acmeplc.com"
    ]
]);

```

Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.redirect_flows.create(params={
  "description": "Wine boxes",
  "session_token": "SESS_wSs0uGYMISxzqOBq",
  "success_redirect_url": "https://example.com/pay/confirm",
  "prefilled_customer": {
    "given_name": "Frank",
    "family_name": "Osborne",
    "email": "frank.osborne@acmeplc.com"
  }
})

```

Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.redirect_flows.create(
  params: {
    description: "Team membership",
    session_token: "my_unique_tracking_id",
    success_redirect_url: "https://example.com/pay/confirm",
    prefilled_customer: {
      given_name: "Frank",
      family_name: "Osborne",
      email: "frank.osborne@acmeplc.com"
    }
  }
)

```

Java

```

import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

RedirectFlow redirectFlow = client
  .withDescription("Wine boxes")
  .withSessionToken("SESS_wSs0uGYMISxzqOBq")
  .withSuccessRedirectUrl("https://example.com/pay/confirm")
  .withPrefilledCustomerGivenName("Frank")
  .withPrefilledCustomerFamilyName("Osborne")
  .withPrefilledCustomerEmail("frank.osborne@acmeplc.com")
  .execute();

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const redirectFlow = await client.redirectFlows.create({
  description: "Wine boxes",
  session_token: "SESS_wSs0uGYMISxzqOBq",
  success_redirect_url: "https://example.com/pay/confirm",
  prefilled_customer: {
    given_name: "Frank",
    family_name: "Osborne",
    email: "frank.osborne@acmeplc.com"
  }
});

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var prefilledCustomer = new GoCardless.Services.RedirectFlowCreateRequest.RedirectFlowPrefilledCustomer()
{
  AddressLine1 = "338 Goswell Road",
  Email = "api@gocardless.com",
  GivenName = "Bobby",
  FamilyName = "Tables"
};

var redirectFlowRequest = new GoCardless.Services.RedirectFlowCreateRequest()
{
  Description = "Gold package",
  SessionToken = "SESS_wSs0uGYMISxzqOBq",
  SuccessRedirectUrl = "https://example.com/pay/confirm",
  PrefilledCustomer = prefilledCustomer
};

var redirectFlowResponse = await gocardless.RedirectFlows.CreateAsync(redirectFlowRequest);
GoCardless.Resources.RedirectFlow redirectFlow = redirectFlowResponse.RedirectFlow;

Console.WriteLine(redirectFlow.RedirectUrl);

Go

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  redirectFlowCreateParams := gocardless.RedirectFlowCreateParams{
    Description: "Cider Barrels",
    Links: &gocardless.RedirectFlowCreateParamsLinks{
      Creditor: "CR00006A7FRJAS",
    },
    PrefilledCustomer: &gocardless.RedirectFlowCreateParamsPrefilledCustomer{
      AddressLine1: "338-346 Goswell Road",
      City:         "London",
      GivenName:   "Tim",
      FamilyName:  "Rogers",
      Email:       "tim@gocardless.com",
      PostalCode:  "EC1V 7LQ",
    },
    Scheme:          "bacs",
    SessionToken:   "dummy_session_token",
    SuccessRedirectUrl: "https://developer.gocardless.com/example-redirect-uri/",
  }

  redirectFlow, err := client.RedirectFlows.Create(ctx, redirectFlowCreateParams)
}

```

### Get a single redirect flow

Returns all details about a single resource.

Relative endpoint: GET /node/nect

**Deprecated:** Redirect Flows are 1.

- HTTP
- PHP
- Python
- Ruby
- Java
- JavaScript
- C/C++

GET https://api.gocardless.com/

```
HTTP/1.1 200 OK
Content-Type: application/json
```

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité

or your payment flows



Consultez notre politique de confidentialité

```
{
  "redirect_flows": {
    "id": "RE123",
    "description": "Wine boxes",
    "session_token": "SESS_wSs0uGYMISxzq0Bq",
    "scheme": null,
    "success_redirect_url": "https://example.com/pay/confirm",
    "redirect_url": "http://pay.gocardless.com/flow/RE123",
    "created_at": "2014-10-22T13:10:06.000Z",
    "links": {
      "creditor": "CR123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->redirectFlows()->get("RE123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.redirect_flows.get("RE123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.redirect_flows.get("RE123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
RedirectFlow redirectFlow = client.redirectFlows.get("RE123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const redirectFlow = await client.redirectFlows.find("RE123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var redirectFlowResponse = await gocardless.RedirectFlows.GetAsync("RE123");
GoCardless.Resources.RedirectFlow redirectFlow = redirectFlowResponse.RedirectFlow;
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  redirectFlow, err := client.P
}
```

## Complete a redirect flow

This creates a [customer](#), [customer](#)

This will return a `redirect_flow`. Your integration has already completed a mandate.

Relative endpoint: POST /redirect/flows/{flow\_id}/complete

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

created mandate.

`flow_already_completed` error if the redirect flow was

**Deprecated:** Redirect Flows are legacy APIs and cannot be used by new integrators. The [Billing Request Flow](#) API should be used for your payment flows.

## Parameters

## session\_token

The customer's session ID must be provided when the redirect flow is set up and again when it is completed. This allows integrators to ensure that the user who was originally sent to the GoCardless payment pages is the one who has completed them.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/redirect_flows/RE123/actions/complete HTTP/1.1
Content-Type: application/json
{
  "data": {
    "session_token": "SESS_wSs0uGYMISxzq0Bq"
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "redirect_flows": {
    "id": "RE123",
    "description": "Wine boxes",
    "session_token": "SESS_wSs0uGYMISxzq0Bq",
    "scheme": null,
    "success_redirect_url": "https://example.com/pay/confirm",
    "confirmation_url": "https://pay.gocardless.com/flow/RE123/success",
    "created_at": "2014-10-22T13:10:06.000Z",
    "mandate_reference": "AB1YZ"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->redirectFlows()->complete("RE123", [
  "params" => ["session_token" => "SESS_wSs0uGYMISxzq0Bq"]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.redirect_flows.complete("RE123", params={
  "session_token": "SESS_wSs0uGYMISxzq0Bq"
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.redirect_flows.complete(
  "RE123",
  params: {
    session_token: "my_unique_tracking_id"
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.redirectFlows().complete("RE123")
  .withSessionToken("SESS_wSs0uGYMISxzq0Bq")
  .execute();
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const redirectFlowResponse = await client.redirectFlows()
  .complete("RE123",
  {
    session_token: "SESS_wSs0uGYMISxzq0Bq"
  })

```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(Environment.SANDBOX)
  .build();
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

```

var redirectFlowRequest = new GoCardless.Services.RedirectFlowCompleteRequest()
{
  SessionToken = "SESS_wSs0uGYMISxzq0Bq"
};

var redirectFlowResponse = await gocardless.RedirectFlows.CompleteAsync("RE123", redirectFlowRequest);
Go

package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }

  redirectFlowCompleteParams := gocardless.RedirectFlowCompleteParams{
    SessionToken: "dummy_session_token",
  }

  redirectFlow, err := client.RedirectFlows.Complete(ctx, "RE0003QNP5DE2101R80QZHJ2X12P93Q4", redirectFlowCompleteParams)
}

```

## Refunds

Refund objects represent (partial) refunds of a [payment](#) back to the [customer](#).

GoCardless will notify you via a [webhook](#) whenever a refund is created, and will update the `amount_refunded` property of the payment.

### Properties

`id` Unique identifier, beginning with “RF”.

`amount` Amount in minor unit (e.g. pence in GBP, cents in EUR).

`created_at` Fixed [timestamp](#), recording when this resource was created.

`currency` [ISO 4217](#) currency code. This is set to the currency of the refund’s [payment](#).

`fx[estimated_exchange_rate]` Estimated rate that will be used in the foreign exchange of the `amount` into the `fx_currency`. This will vary based on the prevailing market rate until the moment that it is paid out. Present only before a resource is paid out. Has up to 10 decimal places.

`fx[exchange_rate]` Rate used in the foreign exchange of the `amount` into the `fx_currency`. Present only after a resource is paid out. Has up to 10 decimal places.

`fx[fx_amount]` Amount that was paid out in the `fx_currency` after foreign exchange. Present only after the resource has been paid out.

`fx[fx_currency]` [ISO 4217](#) code for the currency in which amounts will be paid out (after foreign exchange). Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported. Present only if payouts will be (or were) made via foreign exchange.

`metadata` Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`reference` An optional reference that will appear on your customer’s bank statement. The character limit for this reference is dependent on the scheme.

- ACH** - 10 characters
- Autogiro** - 11 characters
- Bacs** - 10 characters
- BECS** - 30 characters
- BECS NZ** - 12 characters
- Betalingservice** - 30 characters
- Faster Payments** - 18 characters
- PAD** - scheme doesn’t offer references
- PayTo** - 18 characters
- SEPA** - 140 characters

Note that this reference must be unique (for each merchant) for the BECS scheme as it is a scheme requirement. **Restricted:** You can only specify a payment reference for Bacs payments (that is, when collecting from the UK) if you’re on the [GoCardless Plus, Pro or Enterprise packages](#). **Restricted:** You can not specify a payment reference for Autogiro, Faster Payments, PAD or PayTo.

`status` One of:

- `created`: the refund has been initiated
- `pending_submission`: the refund has been submitted to the payment provider
- `submitted`: the refund has been submitted to the payment provider and is awaiting confirmation
- `paid`: the refund has been paid
- `cancelled`: the refund has been cancelled
- `bounced`: the refund has been bounced

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- `funds_returned`: the refund has had its funds returned
- `links[mandate]`  
ID of the [mandate](#) against which the refund is being made.
- `links[payment]`  
ID of the [payment](#) against which the refund is being made.

## Create a refund

Creates a new refund object.

This fails with:

- `total_amount_confirmation_invalid` if the confirmation amount doesn't match the total amount refunded for the payment. This safeguard is there to prevent two processes from creating refunds without awareness of each other.
- `available_refund_amount_insufficient` if the creditor does not have sufficient balance for refunds available to cover the cost of the requested refund.

Relative endpoint: POST /refunds

**Restricted:** This endpoint is disabled by default. To enable it, you will need to enable refunds on your GoCardless Dashboard.

**Warning:** A payment that has been (partially) refunded can still receive a late failure or chargeback from the customer's bank.

A refund can only be created once it is outside the safer refund period (7 calendar days after the charge date) otherwise an error is returned. The safer refund period can be disabled by contacting GoCardless.

Parameters

`amount`  
*required* Amount in minor unit (e.g. pence in GBP, cents in EUR).

`metadata`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

`reference`  
An optional reference that will appear on your customer's bank statement. The character limit for this reference is dependent on the scheme.

**ACH** - 10 characters

**Autogiro** - 11 characters

**Bacs** - 10 characters

**BECS** - 30 characters

**BECS NZ** - 12 characters

**Betalingservice** - 30 characters

**Faster Payments** - 18 characters

**PAD** - scheme doesn't offer references

**PayTo** - 18 characters

**SEPA** - 140 characters

Note that this reference must be unique (for each merchant) for the BECS scheme as it is a scheme requirement. **Restricted:** You can only specify a payment reference for Bacs payments (that is, when collecting from the UK) if you're on the [GoCardless Plus, Pro or Enterprise packages](#). **Restricted:** You can not specify a payment reference for Faster Payments.

`total_amount_confirmation`

Total expected refunded amount in minor unit (e.g. pence/cents/öre). If there are other partial refunds against this payment, this value should be the sum of the existing refunds plus the amount of the refund being created.

Must be supplied if `links[payment]` is present. It is possible to opt out of requiring `total_amount_confirmation`, please contact [our support team](#) for more information.

`links[mandate]`

ID of the [mandate](#) against which the refund is being made.

**Restricted:** You must request access to Mandate Refunds by contacting [our support team](#).

`links[payment]`

ID of the [payment](#) against which the refund is being made.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST <https://api.gocardless.com/refunds> HTTP/1.1

Content-Type: application/json

```
{
  "refunds": {
    "amount": 100,
    "total_amount_confirmation": 150,
    "reference": "Acme refund",
    "metadata": {
      "reason": "late delivery"
    },
    "links": {
      "payment": "PM123"
    }
  }
}
```

HTTP/1.1 201 Created  
Location: /payments/RF123

Content-Type: application/json

```
{
  "refunds": {
    "id": "RF123",
    "created_at": "2014-05-08T",
    "amount": 100,
    "currency": "GBP",
    "reference": "Acme refund",
    "fx": {
      "fx_currency": "EUR",
      "fx_amount": null,
      "exchange_rate": null,
      "estimated_exchange_rate": "1.1234567890"
    }
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        },
        "status": "created",
        "metadata": {
          "reason": "late delivery"
        },
        "links": {
          "payment": "PM123"
        }
      }

// Or, if the creditor does not have a sufficient balance to create the refund:
HTTP/1.1 422 Validation Failed
Content-Type: application/json
{
  "error": {
    "message": "It's not possible to process this refund because you don't have enough funds in GBP. Your available refund amount for payments in GBP",
    "errors": [
      {
        "reason": "available_refund_amount_insufficient",
        "message": "It's not possible to process this refund because you don't have enough funds in GBP. Your available refund amount for payments in GBP",
        "metadata": {
          "available_refund_amount": 90
        }
      }
    ],
    "documentation_url": "https://developer.gocardless.com/api-reference#available_refund_amount_insufficient",
    "type": "validation_failed",
    "request_id": "...",
    "code": 422
  }
}
PHP

```

```

$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->refunds()->create([
  'params' => [
    "amount" => 100,
    "total_amount_confirmation" => 150,
    "reference" => "Acme refund",
    "metadata" => [
      "reason" => "Late delivery"
    ],
    "links" => [
      "payment" => "PM123"
    ]
]);

```

## Python

```

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.refunds.create(params={
  "amount": 100,
  "total_amount_confirmation": 150,
  "reference": "Acme refund",
  "metadata": {
    "reason": "late delivery"
  },
  "links": {
    "payment": "PM123"
})

```

## Ruby

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.refunds.create(
  params: {
    amount: 100,
    total_amount_confirmation: 150,
    reference: "Service refund",
    metadata: { reason: "Late delivery" },
    links: { payment: "PM123" }
)

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.refunds().create()
  .withAmount(100)
  .withTotalAmountConfirmation(150)
  .withReference("Acme refund")
  .withMetadata("reason", "late delivery")
  .withLinksPayment("PM123")
  .execute();

```

## JavaScript

```

const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
const refund = await client.refunds.create({
  amount: 100,
  total_amount_confirmation: 150,
  reference: "Acme refund",
  metadata: {
    reason: "late delivery"
  },
  links: {
    payment: "PM123"
  }
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var refundRequest = new GoCardless.Services.RefundCreateRequest()
{
  Amount = 100,
  TotalAmountConfirmation = 150,
  Reference = "Partial refund",
  Links = new GoCardless.Services.RefundCreateRequest.RefundLinks()
  {
    Payment = "PM0123"
  }
};
var refundResponse = await gocardless.Refunds.CreateAsync(refundRequest);
GoCardless.Resources.Refund refund = refundResponse.Refund;
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Println("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    refundCreateParams := gocardless.RefundCreateParams{
        Amount:                100,
        TotalAmountConfirmation: 150,
        Reference:             "Acme refund",
        Metadata:              map[string]interface{}{"reason": "late delivery"},
        Links:                 gocardless.RefundCreateParamsLinks{
            Payment: "PM123",
        },
    }

    refund, err := client.Refunds.Create(context, refundCreateParams)
}
```

## List refunds

Returns a [cursor-paginated](#) list of your refunds.

Relative endpoint: GET /refunds

## Parameters

**after** Cursor pointing to the start of the desired set.

before

Cursor pointing to the end of the desired set.  
created\_at[gt]

Limit to records created after the specified date-time.  
`created_at[gte]`

Limit to records created on or after the specified date-time.

`created_at[lt]` Limit to records created before the specified date/time.

**created\_at[lt=]** Limit to records created before the specified date-time.

`created_at[lt]` Limit to records created on

**Nous utilisons des limites**

Number of records to return

mandate Les cookies nous permet

Unique identifier, beginning fonctionnement de notre  
with a single character.  
système.

produits ainsi que de vous offrir une expérience de navigation sans précédent.

Unique identifier, beginning  
no sound, type

**refund\_type** Whether a refund was issued or not.

Whether a refund was issued or not, the company must respect the consumer's right to withdraw from the contract within 14 days of receiving the goods.

- payment: *default* return publicite.

Digitized by srujanika@gmail.com

- mandate: returns refund

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
GET https://api.gocardless.com/refunds HTTP/1.1
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "refunds": [
    {
      "id": "RF123",
      "created_at": "2014-05-08T17:01:06.000Z",
      "amount": 100,
      "currency": "GBP",
      "reference": "Acme refund",
      "fx": {
        "fx_currency": "EUR",
        "fx_amount": null,
        "exchange_rate": null,
        "estimated_exchange_rate": "1.1234567890"
      },
      "status": "created",
      "metadata": {
        "reason": "late failure"
      },
      "links": {
        "payment": "PM123"
      }
    }
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->refunds()->list();

$client->refunds()->list([
  'params' => ["payment" => "PM123"]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.refunds.list()

client.refunds.list(params={"payment": "PM123"}).records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.refunds.list

@client.refunds.list(params: { payment: "PM123" })

@client.refunds.list.records.each { |refund| puts refund.inspect }
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

for (Refund refund : client.refunds().all().withPayment("PM123").execute()) {
  System.out.println(refund.getId());
}
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const refunds = await client.refunds().list();

// List all refunds associated with a payment
const refunds = await client.refunds().list({
  payment: "PM123"
});

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(Environment.SANDBOX)
  .build();

var refundRequest = new GoCardlessClient.RefundRequest()
{
  payment_mandate_id: "MD0123"
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

};

var refundListResponse = gocardless.Refunds.All(refundRequest);
foreach (GoCardless.Resources.Refund refund in refundListResponse)
{
    Console.WriteLine(refund.Id);
}
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    refundListParams := gocardless.RefundListParams{
        Mandate: "MD123",
    }
    refundListResult, err := client.Refunds.List(context, refundListParams)
    for _, refund := range refundListResult.Refunds {
        fmt.Println(refund.Id)
    }
}

```

## Get a single refund

Retrieves all details for a single refund

Relative endpoint: GET /refunds/RF123

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/refunds/RF123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "refunds": {
        "id": "RF123",
        "created_at": "2014-05-08T17:01:06.000Z",
        "amount": 100,
        "currency": "GBP",
        "reference": "Acme refund",
        "fx": {
            "fx_currency": "EUR",
            "fx_amount": null,
            "exchange_rate": null,
            "estimated_exchange_rate": "1.1234567890"
        },
        "status": "created",
        "metadata": {
            "reason": "late failure"
        },
        "links": {
            "payment": "PM123"
        }
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->refunds()->get("RF123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client()
```

```
client.refunds.get("RF123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
    access_token: "your_access_token_here",
    environment: :sandbox
)
```

```
@client.refunds.get("RF123")
```

Java

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

Refund refund = client.refunds().get("RF123").execute();
JavaScript
```

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);
```

```
const refund = await client.refunds.find("RF123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);
```

```
var refundResponse = await gocardless.Refunds.GetAsync("RF0123");
GoCardless.Resources.Refund refund = refundResponse.Refund;
```

Go

```
package main
```

```
import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }
    refund, err := client.Refunds.Get(context, "RF123")
}
```

## Update a refund

Updates a refund object.

Relative endpoint: PUT /refunds/RF123

Parameters

`metadata`

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
PUT https://api.gocardless.com/refunds/RF123 HTTP/1.1
Content-Type: application/json
```

```
{
    "refunds": {
        "metadata": {
            "key": "value"
        }
    }
}
```

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "refunds": {
        "id": "RF123",
        "created_at": "2014-05-08T17:01:06.000Z",
        "amount": 100,
        "currency": "GBP",
        "reference": "Acme refund",
        "fx": {
            "fx_currency": "EUR",
            "fx_amount": null,
            "exchange_rate": null,
            "estimated_exchange_rate": null
        },
        "status": "created",
        "metadata": {
            "key": "value"
        },
        "links": {
            "payment": "PM123"
        }
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->refunds()->update("RF123", [
    'params' => ["metadata" => ["key" => "value"]]
]);

```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.refunds.update("RF123", params={
    "metadata": {"key": "value"}
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
    access_token: "your_access_token",
    environment: :sandbox
)

@client.refunds.update(
    "RF123",
    params: {
        metadata: { reason: "Late delivery", internal_code: "refund_1A" }
    }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.refunds().update("RF123")
    .withMetadata("internal_code", "refund_1A")
    .execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const refund = await client.refunds.update(
    "RF123",
    {
        metadata: {
            key: "value"
        }
    }
);
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var refundRequest = new GoCardless.Services.RefundUpdateRequest()
{
    Metadata = new Dictionary<string, string>()
    {
        {"reason", "Late delivery"}
    }
};

var refundResponse = await gocardless.Refunds.UpdateAsync("RF0123", refundRequest);
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.NewClient(config)
    if err != nil {
        fmt.Println("error in init")
        return
    }

    refundUpdateParams := gocardless.Metadata{map[string]interface{}{"key": "value"}}
    refund, err := client.Refund(
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**Scenario Simulators**

Scenario Simulators allow you to manually trigger and test certain paths that your integration will encounter in the real world. These endpoints are only active in the sandbox environment.

## Properties

### id

The unique identifier of the simulator, used to initiate simulations. One of:

- `creditor_verification_status_action_required`: Sets a creditor's verification status to `action required`, meaning that the creditor must provide further information to GoCardless in order to verify their account to receive payouts.
- `creditor_verification_status_in_review`: Sets a creditor's verification status to `in review`, meaning that the creditor has provided all of the information requested by GoCardless to verify their account, and is now awaiting review.
- `creditor_verification_status_successful`: Sets a creditor's verification status to `successful`, meaning that the creditor is fully verified and can receive payouts.
- `payment_confirmed`: Transitions a payment through to `confirmed`. It must start in the `pending_submission` state, and its mandate must be in the `activated` state (unless it is a payment for ACH, BECS, BECS\_NZ or SEPA, in which cases the mandate may be `pending_submission`, since their mandates are submitted with their first payment).
- `payment_paid_out`: Transitions a payment through to `paid_out`, having been collected successfully and paid out to you. It must start in the `pending_submission` state, and its mandate must be in the `activated` state (unless it is a payment for ACH, BECS, BECS\_NZ or SEPA, in which cases the mandate may be `pending_submission`, since their mandates are submitted with their first payment).
- `payment_failed`: Transitions a payment through to `failed`. It must start in the `pending_submission` state, and its mandate must be in the `activated` state (unless it is a payment for ACH, BECS, BECS\_NZ or SEPA, in which cases the mandate may be `pending_submission`, since their mandates are submitted with their first payment).
- `payment_charged_back`: Behaves the same as the `payout_paid_out` simulator, except that the payment is transitioned to `charged_back` after it is paid out, having been charged back by the customer.
- `payment_chargeback_settled`: Behaves the same as the `payment_charged_back` simulator, except that the charged back payment is additionally included as a debit item in a payout, thereby settling the charged back payment.
- `payment_late_failure`: Transitions a payment through to `late_failure`, having been apparently collected successfully beforehand. It must start in either the `pending_submission` or `paid_out` state, and its mandate must be in the `activated` state (unless it is a payment for ACH, BECS, BECS\_NZ or SEPA, in which cases the mandate may be `pending_submission`, since their mandates are submitted with their first payment). Not compatible with Autogiro mandates.
- `payment_late_failure_settled`: Behaves the same as the `payment_late_failure` simulator, except that the late failure is additionally included as a debit item in a payout, thereby settling the late failure.
- `payment_submitted`: Transitions a payment to `submitted`, without proceeding any further. It must start in the `pending_submission` state.
- `mandate_activated`: Transitions a mandate through to `activated`, having been submitted to the banks and set up successfully. It must start in the `pending_submission` state. Not compatible with ACH, BECS, BECS\_NZ and SEPA mandates, which are submitted and activated with their first payment.
- `mandate_customer_approval_granted`: Transitions a mandate through to `pending_submission`, as if the customer approved the mandate creation. It must start in the `pending_customer_approval` state. Compatible only with Bacs and SEPA mandates, which support customer signatures on the mandate. All payments associated with the mandate will be transitioned to `pending_submission`. All subscriptions associated with the mandate will become `active`.
- `mandate_customer_approval_skipped`: Transitions a mandate through to `pending_submission`, as if the customer skipped the mandate approval during the mandate creation process. It must start in the `pending_customer_approval` state. Compatible only with Bacs and SEPA mandates, which support customer signatures on the mandate. All payments associated with the mandate will be transitioned to `pending_submission`. All subscriptions associated with the mandate will become `active`.
- `mandate_failed`: Transitions a mandate through to `failed`, having been submitted to the banks but found to be invalid (for example due to invalid bank details). It must start in the `pending_submission` or `submitted` states. Not compatible with SEPA mandates, which are submitted with their first payment.
- `mandate_expired`: Transitions a mandate through to `expired`, having been submitted to the banks, set up successfully and then expired because no collection attempts were made against it for longer than the scheme's dormancy period (13 months for Bacs, 3 years for SEPA, 15 months for ACH, Betalingsservice, and BECS). It must start in the `pending_submission` state. Not compatible with Autogiro, BECS NZ, and PAD mandates, which do not expire.
- `mandate_transferred`: Transitions a mandate through to `transferred`, having been submitted to the banks, set up successfully and then moved to a new bank account due to the customer using the UK's Current Account Switching Service (CASS). It must start in the `pending_submission` state. Only compatible with Bacs mandates.
- `mandate_transferred_with_resubmission`: Transitions a mandate through `transferred` and resubmits it to the banks, can be caused by the UK's Current Account Switching Service (CASS) or when a customer contacts GoCardless to change their bank details. It must start in the `pending_submission` state. Only compatible with Bacs mandates.
- `mandate_suspended_by_payer`: Transitions a mandate to `suspended_by_payer`, as if payer has suspended the mandate after it has been setup successfully. It must start in the `activated` state. Only compatible with PAV\_TO mandates.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

ysettling the funds. It must start

, with all actions completed

- `billing_request_fulfilled_and_payment_failed`: Authorises the billing request, fulfils it, and moves the associated payment to `failed`. The billing request must be in the pending state, with all actions completed except for `bank_authorisation`. Only billing requests with a `payment_request` are supported.
  - `billing_request_fulfilled_and_payment_confirmed_to_failed`: Authorises the billing request, fulfils it, moves the associated payment to `confirmed` and then moves it to `failed`. The billing request must be in the pending state, with all actions completed except for `bank_authorisation`. Only billing requests with a `payment_request` are supported.
  - `billing_request_fulfilled_and_payment_paid_out`: Authorises the billing request, fulfils it, and moves the associated payment to `paid_out`. The billing request must be in the pending state, with all actions completed except for `bank_authorisation`. Only billing requests with a `payment_request` are supported.

## Simulate a scenario

Runs the specific scenario simulator against the specific resource

Relative endpoint: POST /scenario\_simulators/payment\_failed/actions/run

### Parameters

links[resource]

ID of the resource to run the simulation against. Must be same type of resource as the simulator that is being run. eg. Payment ID for payment\_failed, Mandate ID for mandate\_activated etc

A horizontal row of eight circles. The first circle is filled with a solid blue color, while the remaining seven circles are unfilled with a thin black outline.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/scenario_simulators/payment_failed/actions/run HTTP/1.1
Content-Type: application/json
```

```
content-type: application/json
```

"data": {

}

{ }

```
PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->scenarioSimulators()->run("payment_failed", [
    "params" => ["links" => ["resource" => "PM123"]]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.scenario_simulators.run("payment_failed", params={
    "links": { "resource": "PM123" }
})
```

Ruby

```
@client = GoCardlessPro::Client.new(  
  access_token: "your_access_token",  
  environment: :sandbox  
)
```

```
@client.scenario_simulators.run(  
    "payment_failed",  
    params: { links: { resource: "PM123" } })
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.scenario_simulators().run()
    .withLinks("resource", "PM12")
    .execute();
```

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

.NET

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var simulateRequest = new GoCardless.Services.ScenarioSimulatorRunRequest()
{
    Links = new Dictionary<string, string>()
    {
        {"resource", "PM123"}
    }
};

var response = await gocardless.ScenarioSimulators.RunAsync(
    "payment_failed", simulateRequest
);

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    scenarioSimulatorRunParams := gocardless.ScenarioSimulatorRunParams{
        Links: &gocardless.ScenarioSimulatorRunParamsLinks{
            Resource: "PM123",
        },
    }

    action := "payment_failed"
    scenarioSimulator, err := client.ScenarioSimulators.Run(context, action, scenarioSimulatorRunParams)
}

```

## Scheme Identifiers

This represents a scheme identifier (e.g. a SUN in Bacs or a CID in SEPA). Scheme identifiers are used to specify the beneficiary name that appears on customers' bank statements.

### Properties

<code>id</code>	Unique identifier, usually beginning with “SU”.
<code>address_line1</code>	The first line of the scheme identifier’s support address.
<code>address_line2</code>	The second line of the scheme identifier’s support address.
<code>address_line3</code>	The third line of the scheme identifier’s support address.
<code>can_specify_mandate_reference</code>	Whether a custom reference can be submitted for mandates using this scheme identifier.
<code>city</code>	The city of the scheme identifier’s support address.
<code>country_code</code>	<a href="#">ISO 3166-1 alpha-2 code</a> .
<code>created_at</code>	Fixed <a href="#">timestamp</a> , recording when this resource was created.
<code>currency</code>	The currency of the scheme identifier.
<code>email</code>	Scheme identifier’s support email address.
<code>minimum_advance_notice</code>	The minimum interval, in working days, between the sending of a pre-notification to the customer, and the charge date of a payment using this scheme identifier.

By default, GoCardless sends:

<code>name</code>	The name which appears on the document.
<code>phone_number</code>	Scheme identifier’s support phone number.
<code>postal_code</code>	The scheme identifier’s support postal code.
<code>reference</code>	The scheme-unique identifier.
<code>region</code>	The scheme identifier’s support region.
<code>scheme</code>	The scheme which this scheme identifier belongs to.
<code>status</code>	The status of the scheme identifier.

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

The status of the scheme identifier. Only active scheme identifiers will be applied to a creditor and used against payments.

## Create a scheme identifier

Creates a new scheme identifier. The scheme identifier status will be pending while GoCardless is processing the request. Once the scheme identifier is ready to be used the status will be updated to active. At this point, GoCardless will emit a scheme identifier activated event via webhook to notify you of this change. In Bacs, it will take up to five working days for a scheme identifier to become active. On other schemes, including SEPA, this happens instantly.

### Scheme identifier name validations

The name field of a scheme identifier can contain alphanumeric characters, spaces and special characters.

Its maximum length and the special characters it supports depend on the scheme:

scheme	maximum length	special characters allowed
bacs	18 characters	/ . & -
sepa	70 characters	/ ?:() . , + & <> ' "
ach	16 characters	/ ?:() . , '+ -
faster_payments	18 characters	/ ?:() . , '+ -

The validation error that gets returned for an invalid name will contain a suggested name in the metadata that is guaranteed to pass name validations.

You should ensure that the name you set matches the legal name or the trading name of the creditor, otherwise, there is an increased risk of chargeback.

Relative endpoint: POST /scheme\_identifiers

**Restricted:** This endpoint is restricted to GoCardless Embed customers. Please [contact us](#) if you are interested in using this product.

Parameters

name  
*required* The name which appears on customers' bank statements. This should usually be the merchant's trading name.  
scheme  
*required* The scheme which this scheme identifier applies to.  
links[creditor]  
*required* ID of the associated [creditor](#).



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/scheme_identifiers HTTP/1.1
Content-Type: application/json
{
  "scheme_identifiers": {
    "name": "The Wine Club",
    "scheme": "bacs",
    "links": {
      "creditor": "CR123"
    }
  }
}
HTTP/1.1 201 Created
Location: /scheme_identifiers/SU123
Content-Type: application/json
{
  "scheme_identifiers": {
    "id": "SU123",
    "created_at": "2021-01-23T13:44:19.006Z",
    "name": "The Wine Club",
    "scheme": "bacs",
    "reference": null,
    "status": "pending",
    "minimum_advance_notice": 3,
    "can_specify_mandate_reference": false,
    "currency": "GBP",
    "address_line1": null,
    "address_line2": null,
    "address_line3": null,
    "city": null,
    "region": null,
    "postal_code": null,
    "country_code": null,
    "email": null,
    "phone_number": null
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->schemeIdentifiers()->
  'params' => [
    'scheme' => 'bacs',
    'name' => 'Example',
    'links' => ['cre...
```

Python

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.scheme_identifiers.create(params={
    "scheme": "bacs",
    "name": "Example Ltd",
    "links": {
        "creditor": "CR123"
    }
})
Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.scheme_identifiers.create(
  params: {
    scheme: "bacs",
    name: "Example Ltd",
    links: {
      creditor: "CR123"
    }
  }
)

```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

SchemeIdentifier schemeIdentifier = client.schemeIdentifiers().create()
    .withScheme(SchemeIdentifierService.SchemeIdentifierCreateRequest.Scheme.BACS)
    .withName("The Wine Club")
    .withLinksCreditor("CR123")
    .execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const schemeIdentifier = await client.schemeIdentifiers().create({
  scheme: "bacs",
  name: "The Wine Club",
  links: {
    creditor: "CR123"
  }
});
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

SchemeIdentifierCreateRequest schemeIdentifierRequest = new SchemeIdentifierCreateRequest
{
    Name = "The Wine Club",
    Scheme = SchemeIdentifierCreateRequest.SchemeIdentifierScheme.Bacs,
    Links = new SchemeIdentifierCreateRequest.SchemeIdentifierLinks()
    {
        Creditor = "CR123"
    }
};
```

```
var schemeIdentifierResponse = await client.SchemeIdentifiers.CreateAsync(schemeIdentifierRequest);
GoCardless.Resources.SchemeIdentifier schemeIdentifier = schemeIdentifierResponse.SchemeIdentifier;
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in init\n")
        return
    }
    client, err := gocardless.NewClient(config)
    if err != nil {
        fmt.Println("error in init\n")
        return
    }

    context := context.Background()
    schemeIdentifierCreateParams := gocardless.SchemeIdentifierCreateParams{
        Name: "Durian Co",
        Scheme: "bacs",
        Links: gocardless.SchemeIdentifierLinks{
            Creditor: "CR123",
        },
    }
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

schemeIdentifier, err := client.SchemeIdentifiers.Create(context, schemeIdentifierCreateParams)
if err != nil {
    fmt.Printf("error creating scheme identifier: %s", err.Error())
    return
}
}

```

## List scheme identifiers

Returns a [cursor-paginated](#) list of your scheme identifiers.

Relative endpoint: GET /scheme\_identifiers

Parameters

**after**  
Cursor pointing to the start of the desired set.  
**before**  
Cursor pointing to the end of the desired set.  
**creditor**  
Unique identifier, beginning with “CR”.  
**limit**  
Number of records to return.

○ ○ ○ ○ ○ ○ ○ ○

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/scheme\_identifiers HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "scheme_identifiers": [
    {
      "id": "SU123",
      "created_at": "2021-01-23T09:41:36.234Z",
      "name": "The Wine Club",
      "scheme": "bacs",
      "reference": null,
      "status": "active",
      "minimum_advance_notice": 3,
      "can_specify_mandate_reference": false,
      "currency": "GBP",
      "address_line1": null,
      "address_line2": null,
      "address_line3": null,
      "city": null,
      "region": null,
      "postal_code": null,
      "country_code": null,
      "email": null,
      "phone_number": null
    }
  ],
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->schemeIdentifiers()->list()->records;
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.scheme_identifiers.list().records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.scheme_identifiers.list
```

Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
for (SchemeIdentifier schemeIdentifier: client.schemeIdentifiers().all().execute()) {
```

```

System.out.println(schemeIdentifier.getId());
}
JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const schemeIdentifierList = await client.schemeIdentifiers.list();

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var schemeIdentifierListResponse = client.SchemeIdentifiers.All();

foreach (GoCardless.Resources.SchemeIdentifier schemeIdentifier in schemeIdentifierListResponse)
{
    Console.WriteLine(schemeIdentifier.Id);
}

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    context := context.Background()
    schemeIdentifierListParams := gocardless.SchemeIdentifierListParams{}

    schemeIdentifierListResult, err := client.SchemeIdentifiers.List(context, schemeIdentifierListParams)
    if err != nil {
        fmt.Printf("error listing scheme identifiers: %s", err.Error())
        return
    }
    for _, schemeIdentifier := range schemeIdentifierListResult.SchemeIdentifiers {
        fmt.Println(schemeIdentifier.Id)
    }
}
}

```

## Get a single scheme identifier

Retrieves the details of an existing scheme identifier.

Relative endpoint: GET /scheme\_identifiers/SU123



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/scheme\_identifiers/SU123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "scheme_identifiers": {
    "id": "SU123",
    "created_at": "2021-01-23T09:41:36.234Z",
    "name": "The Wine Club",
    "scheme": "bacs",
    "reference": null,
    "status": "active",
    "minimum_advance_notice": 3,
    "can_specify_mandate_reference": false,
    "currency": "GBP",
    "address_line1": null,
    "address_line2": null,
    "address_line3": null,
    "city": null,
    "region": null,
    "postal_code": null,
    "country_code": null,
    "email": null,
    "phone_number": null
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
'access_token' => 'your_access_token',
'environment' => \GoCardless\Environment::SANDBOX
]);
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.scheme_identifiers.get("SU123").attributes
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.scheme_identifiers.get("SU123").attributes
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

SchemeIdentifier schemeIdentifier = client.schemeIdentifiers().get("SU123").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const schemeIdentifier = await client.schemeIdentifiers.find("SU123");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var schemeIdentifierResponse = await client.SchemeIdentifiers.GetAsync("SU123");

GoCardless.Resources.SchemeIdentifier schemeIdentifier = schemeIdentifierResponse.SchemeIdentifier;
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    context := context.Background()
    schemeIdentifier, err := client.SchemeIdentifiers.Get(context, "SU123")
    if err != nil {
        fmt.Printf("error getting scheme identifier: %s", err.Error())
        return
    }
}
```

## Subscriptions

Subscriptions create [payments](#) according to a schedule.

### Recurrence Rules

The following rules apply when specifying recurrence:

- If `day_of_month` and `start_date` are not provided `start_date` will be the [mandate](#)'s `next_possible_charge_date` and the subscription will then recur based on the `interval` & `interval_unit`
- If `month` or `day_of_month` are

**interval\_unit month**

yearly	optional (required)
monthly	invalid
weekly	invalid

Examples:

**interval\_unit interval month day**

yearly	1	january	-1	
monthly	6			
monthly	6		12	valid

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

interval_unit	interval	month	day_of_month	valid?
weekly	2			valid
yearly	1	march		invalid - missing day_of_month
yearly	1		2	invalid - missing month
monthly	6	august	12	invalid - month must be blank
weekly	2	october	10	invalid - month and day_of_month must be blank

## Rolling dates

When a charge date falls on a non-business day, one of two things will happen:

- if the recurrence rule specified -1 as the day\_of\_month, the charge date will be rolled **backwards** to the previous business day (i.e., the last working day of the month).
- otherwise the charge date will be rolled **forwards** to the next business day.

### Properties

<b>id</b>	Unique identifier, beginning with “SB”.
<b>amount</b>	Amount in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
<b>app_fee</b>	The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
<b>count</b>	The total number of payments that should be taken by this subscription.
<b>created_at</b>	Fixed <a href="#">timestamp</a> , recording when this resource was created.
<b>currency</b>	<a href="#">ISO 4217</a> currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.
<b>day_of_month</b>	As per RFC 2445. The day of the month to charge customers on. 1-28 or -1 to indicate the last day of the month.
<b>earliest_charge_date_after_resume</b>	The earliest date that will be used as a charge_date on payments created for this subscription if it is resumed. Only present for paused subscriptions. This value will change over time.
<b>end_date</b>	Date on or after which no further payments should be created. If this field is blank and count is not specified, the subscription will continue forever. <b>Deprecated:</b> This field will be removed in a future API version. Use count to specify a number of payments instead.
<b>interval</b>	Number of interval_units between customer charge dates. Must be greater than or equal to 1. Must result in at least one charge date per year. Defaults to 1.
<b>interval_unit</b>	The unit of time between customer charge dates. One of weekly, monthly or yearly.
<b>metadata</b>	Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.
<b>month</b>	Name of the month on which to charge a customer. Must be lowercase. Only applies when the interval_unit is yearly.
<b>name</b>	Optional name for the subscription. This will be set as the description on each payment created. Must not exceed 255 characters.
<b>parent_plan_paused</b>	Whether the parent plan of this subscription is paused.
<b>payment_reference</b>	An optional payment reference. This will be set as the reference on each payment created and will appear on your customer’s bank statement. See the documentation for the <a href="#">create payment endpoint</a> for more details. <b>Restricted:</b> You need your own Service User Number to specify a payment reference for Bacs payments.
<b>retry_if_possible</b>	On failure, automatically retry payments using <a href="#">intelligent retries</a> . Default is false. <b>Important:</b> To be able to use intelligent retries, Success+ needs to be enabled in <a href="#">GoCardless dashboard</a> .
<b>start_date</b>	The date on which the first payment should be charged. Must be on or after the <a href="#">mandate</a> ’s next_possible_charge_date. When left blank and month or day_of_month are provided, this will be set to the date of the first payment. If created without month or day_of_month this will be set as the mandate’s next_possible_charge_date
<b>status</b>	One of: <ul style="list-style-type: none"> <li>pending_customer_approval: the subscription is waiting for customer approval before becoming active</li> <li>customer_approval_denied: the customer did not approve the subscription</li> <li>active: the subscription is currently active and will continue to create payments</li> <li>finished: all of the payments have been completed</li> <li>cancelled: the subscription has been cancelled</li> <li>paused: the subscription has been paused</li> </ul>
<b>upcoming_payments</b>	Up to 10 upcoming payments.
	Each instance will contain the following fields: <ul style="list-style-type: none"> <li>amount: The amount of the payment.</li> <li>charge_date: The date of the payment.</li> </ul>

### Nous utilisons des cookies

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**links[mandate]**

ID of the associated [mandate](#) which the subscription will create payments against.

**Create a subscription**

Creates a new subscription object

Relative endpoint: POST /subscriptions

Parameters

**amount**  
*required* Amount in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**currency**  
*required* [ISO 4217](#) currency code. Currently “AUD”, “CAD”, “DKK”, “EUR”, “GBP”, “NZD”, “SEK” and “USD” are supported.

**interval\_unit**  
*required* The unit of time between customer charge dates. One of `weekly`, `monthly` or `yearly`.

**app\_fee**  
The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).

**count**  
The total number of payments that should be taken by this subscription.

**day\_of\_month**  
As per RFC 2445. The day of the month to charge customers on. 1-28 or -1 to indicate the last day of the month.

**end\_date**  
Date on or after which no further payments should be created.  
If this field is blank and count is not specified, the subscription will continue forever.  
**Deprecated:** This field will be removed in a future API version. Use count to specify a number of payments instead.

**interval**  
Number of interval\_units between customer charge dates. Must be greater than or equal to 1. Must result in at least one charge date per year. Defaults to 1.

**metadata**  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

**month**  
Name of the month on which to charge a customer. Must be lowercase. Only applies when the interval\_unit is `yearly`.

**name**  
Optional name for the subscription. This will be set as the description on each payment created. Must not exceed 255 characters.

**payment\_reference**  
An optional payment reference. This will be set as the reference on each payment created and will appear on your customer’s bank statement. See the documentation for the [create payment endpoint](#) for more details.  
**Restricted:** You need your own Service User Number to specify a payment reference for Bacs payments.

**retry\_if\_possible**  
On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

**start\_date**  
The date on which the first payment should be charged. Must be on or after the [mandate](#)’s next\_possible\_charge\_date. When left blank and month or day\_of\_month are provided, this will be set to the date of the first payment. If created without month or day\_of\_month this will be set as the mandate’s next\_possible\_charge\_date

**links[mandate]**  
*required* ID of the associated [mandate](#) which the subscription will create payments against.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST <https://api.gocardless.com/subscriptions> HTTP/1.1  
Content-Type: application/json

```
{
  "subscriptions": {
    "amount": "2500",
    "currency": "GBP",
    "name": "Monthly Magazine",
    "interval_unit": "monthly",
    "day_of_month": "1",
    "metadata": {
      "order_no": "ABCD1234"
    },
    "links": {
      "mandate": "MA123"
    }
  }
}
```

HTTP/1.1 201 Created  
Location: /subscriptions/SB123  
Content-Type: application/json

```
{
  "subscriptions": {
    "id": "SB123",
    "created_at": "2014-10-20T",
    "amount": 2500,
    "currency": "GBP",
    "status": "active",
    "name": "Monthly Magazine",
    "start_date": "2014-11-03",
    "end_date": null,
    "interval": 1,
    "interval_unit": "monthly",
    "day_of_month": 1,
    "month": null,
    "payment_reference": null,
    "app_fee": null,
    "earliest_charge_date_after": null,
    "upcoming_payments": [
      {"charge_date": "2014-11-03", "amount": 2500}
    ]
  }
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d’optimiser le fonctionnement de notre site, d’améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
{
  "charge_date": "2014-12-01", "amount": 2500 },
  {"charge_date": "2015-01-02", "amount": 2500 },
  {"charge_date": "2015-02-02", "amount": 2500 },
  {"charge_date": "2015-03-02", "amount": 2500 },
  {"charge_date": "2015-04-01", "amount": 2500 },
  {"charge_date": "2015-05-01", "amount": 2500 },
  {"charge_date": "2015-06-01", "amount": 2500 },
  {"charge_date": "2015-07-01", "amount": 2500 },
  {"charge_date": "2015-08-03", "amount": 2500 }
],
"metadata": {
  "order_no": "ABCD1234"
},
"links": {
  "mandate": "MA123"
},
"retry_if_possible": false
}
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->subscriptions()->create([
  "params" => ["amount" => 2500,
    "currency" => "GBP",
    "name" => "Monthly Magazine",
    "interval_unit" => "monthly",
    "day_of_month" => 1,
    "metadata" => ["order_no" => "ABCD1234"],
    "links" => ["mandate" => "MA123"]]
]);
```

### Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.subscriptions.create(params={
  "amount": "2500",
  "currency": "GBP",
  "name": "Monthly Magazine",
  "interval_unit": "monthly",
  "day_of_month": "1",
  "metadata": {
    "order_no": "ABCD1234"
  },
  "links": {
    "mandate": "MA123"
  }
})
```

### Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.subscriptions.create(
  params: {
    amount: 2500,
    currency: "GBP",
    name: "Monthly magazine",
    interval_unit: "monthly",
    day_of_month: 1,
    links: {
      mandate: "MD123"
    }
  }
)
```

### Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
import com.gocardless.services.SubscriptionService.SubscriptionCreateRequest.IntervalUnit;
```

```
Subscription subscription = client.subscriptions().create()
  .withAmount(2500)
  .withCurrency("GBP")
  .withName("Monthly Magazine")
  .withIntervalUnit(IntervalUnit.MONTHLY)
  .withDayOfMonth(1)
  .withMetadata("order_no", "ABCD1234")
  .withLinksMandate("MD123")
  .execute();
```

### JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const subscription = await client.subscriptions.create({
  amount: "2500",
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

currency: "GBP",
name: "Monthly Magazine",
interval_unit: "monthly",
day_of_month: "1",
metadata": {
    order_no: "ABCD1234"
},
links: {
    mandate: "MA123"
}
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var subscriptionRequest = new GoCardless.Services.SubscriptionCreateRequest()
{
    Amount = 1000,
    Currency = "GBP",
    Name = "Monthly subscription",
    Interval = 1,
    IntervalUnit = GoCardless.Services.SubscriptionCreateRequest.SubscriptionIntervalUnit.Monthly,
    Links = new GoCardless.Services.SubscriptionCreateRequest.SubscriptionLinks()
    {
        Mandate = "MD0123"
    }
};

var subscriptionResponse = await gocardless.Subscriptions.CreateAsync(subscriptionRequest);
GoCardless.Resources.Subscription subscription = subscriptionResponse.Subscription;

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    subscriptionCreateParams := gocardless.SubscriptionCreateParams{
        Amount:      1500, // 15 GBP in pence
        Currency:    "GBP",
        IntervalUnit: "monthly",
        DayOfMonth:  5,
        Links: gocardless.SubscriptionCreateParamsLinks{
            Mandate: "MD123",
        },
        Metadata: map[string]interface{}{"subscription_number": "ABC123"},
    }

    subscription, err := client.Subscriptions.Create(ctx, subscriptionCreateParams)
}
}

```

## List subscriptions

Returns a [cursor-paginated](#) list of your subscriptions. Please note if the subscriptions are related to customers who have been removed, they will not be shown in the response.

Relative endpoint: GET /subscriptions

### Parameters

**after**  
Cursor pointing to the start of the desired set.

**before**  
Cursor pointing to the end of the desired set.

**created\_at[gt]**  
Limit to records created after the specified date-time.

**created\_at[gte]**  
Limit to records created on or after the specified date-time.

**created\_at[lt]**  
Limit to records created before the specified date-time.

**created\_at[lte]**  
Limit to records created on or before the specified date-time.

**customer**  
Unique identifier, beginning with

**limit**  
Number of records to return.

**mandate**  
Unique identifier, beginning with

**status**  
Upto 5 of:

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- pending\_customer\_approval
- customer\_approval\_denied
- active
- finished
- cancelled
- paused

Omit entirely to include subscriptions in all states.



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET <https://api.gocardless.com/subscriptions> HTTP/1.1

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "meta": {
    "cursors": {
      "before": null,
      "after": null
    },
    "limit": 50
  },
  "subscriptions": [
    {
      "id": "SB123",
      "created_at": "2014-10-20T17:01:06.000Z",
      "amount": 2500,
      "currency": "GBP",
      "status": "active",
      "name": "Monthly Magazine",
      "start_date": "2014-11-03",
      "end_date": null,
      "interval": 1,
      "interval_unit": "monthly",
      "day_of_month": 1,
      "month": null,
      "payment_reference": null,
      "earliest_charge_date_after_resume": null,
      "parent_plan_paused": false,
      "upcoming_payments": [
        { "charge_date": "2014-11-03", "amount": 2500 },
        { "charge_date": "2014-12-01", "amount": 2500 },
        { "charge_date": "2015-01-02", "amount": 2500 },
        { "charge_date": "2015-02-02", "amount": 2500 },
        { "charge_date": "2015-03-02", "amount": 2500 },
        { "charge_date": "2015-04-01", "amount": 2500 },
        { "charge_date": "2015-05-01", "amount": 2500 },
        { "charge_date": "2015-06-01", "amount": 2500 },
        { "charge_date": "2015-07-01", "amount": 2500 },
        { "charge_date": "2015-08-03", "amount": 2500 }
      ],
      "metadata": {
        "order_no": "ABCD1234"
      },
      "links": {
        "mandate": "MA123"
      },
      "retry_if_possible": false
    }
  ]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->subscriptions()->list();

$client->subscriptions()->list([
  'params' => ["customer" => "CU123"]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.subscriptions.list().re
client.subscriptions.list(par
Ruby
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token_here",
  environment: :sandbox
)
@client.subscriptions.list()
@client.subscriptions.list(par
@client.subscriptions.list(reco
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

for (Subscription subscription : client.subscriptions().all().withCustomer("CU123").execute()) {
    System.out.println(subscription.getId());
}
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const subscriptions = await client.subscriptions.list();

// List all subscriptions associated with a given customer.
const subscriptions = await client.subscriptions.list({ customer: "CU123" });
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var subscriptionRequest = new GoCardless.Services.SubscriptionListRequest()
{
    Customer = "CU000123"
};

var subscriptionListResponse = gocardless.Subscriptions.All(subscriptionRequest);
foreach (GoCardless.Resources.Subscription subscription in subscriptionListResponse)
{
    Console.WriteLine(subscription.Id);
}
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    subscriptionListParams := gocardless.SubscriptionListParams{}
    subscriptionListResult, err := client.Subscriptions.List(context, subscriptionListParams)
    for _, subscription := range subscriptionListResult.Subscriptions {
        fmt.Println(subscription.Id)
    }
}
```

**Get a single subscription**

Retrieves the details of a single subscription.

Relative endpoint: GET /subscriptions/SB123

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET <https://api.gocardless.com/subscriptions/SB123> HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "subscriptions": {
        "id": "SB123",
        "created_at": "2014-10-20T14:00:00Z",
        "amount": 2500,
        "currency": "GBP",
        "status": "active",
        "name": "Monthly Magazine",
        "start_date": "2014-11-03",
        "end_date": null,
        "interval": 1,
        "interval_unit": "monthly",
        "day_of_month": 1,
        "month": null,
        "payment_reference": null,
        "earliest_charge_date_after": null,
        "parent_plan_paused": false,
        "upcoming_payments": [
            { "charge_date": "2014-11-03", "amount": 2500 },
            { "charge_date": "2014-12-01", "amount": 2500 }
        ]
    }
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    {
      "charge_date": "2014-12-01", "amount": 2500 },
      "charge_date": "2015-01-02", "amount": 2500 },
      "charge_date": "2015-02-02", "amount": 2500 },
      "charge_date": "2015-03-02", "amount": 2500 },
      "charge_date": "2015-04-01", "amount": 2500 },
      "charge_date": "2015-05-01", "amount": 2500 },
      "charge_date": "2015-06-01", "amount": 2500 },
      "charge_date": "2015-07-01", "amount": 2500 },
      "charge_date": "2015-08-03", "amount": 2500 }
    ],
    "metadata": {
      "order_no": "ABCD1234"
    },
    "links": {
      "mandate": "MA123"
    },
    "retry_if_possible": false
  }
}

```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->subscriptions()->get("SB123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.subscriptions.get("SB123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.subscriptions.get("SB123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
Subscription subscription = client.subscriptions().get("SB123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const subscription = await client.subscriptions.find("SB123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var subscriptionResponse = await gocardless.Subscriptions.GetAsync("SB0123");
GoCardless.Resources.Subscription subscription = subscriptionResponse.Subscription;
```

Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in init")
    return
  }
  subscription, err := client.Subscription("SB123")
}
```

## Update a subscription

Updates a subscription object.

This fails with:

- validation\_failed if invalid data is provided when attempting to update a subscription.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

- `subscription_not_active` if the subscription is no longer active.
- `subscription_already_ended` if the subscription has taken all payments.
- `mandate_payments_require_approval` if the amount is being changed and the mandate requires approval.
- `number_of_subscription_amendments_exceeded` error if the subscription amount has already been changed 10 times.
- `forbidden` if the amount is being changed, and the subscription was created by an app and you are not authenticated as that app, or if the subscription was not created by an app and you are authenticated as an app
- `resource_created_by_another_app` if the app fee is being changed, and the subscription was created by an app other than the app you are authenticated as

Relative endpoint: PUT /subscriptions/SB123

#### Parameters

- `amount`  
Amount in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
- `app_fee`  
The amount to be deducted from each payment as an app fee, to be paid to the partner integration which created the subscription, in the lowest denomination for the currency (e.g. pence in GBP, cents in EUR).
- `metadata`  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.
- `name`  
Optional name for the subscription. This will be set as the description on each payment created. Must not exceed 255 characters.
- `payment_reference`  
An optional payment reference. This will be set as the reference on each payment created and will appear on your customer's bank statement. See the documentation for the [create payment endpoint](#) for more details.  
**Restricted:** You need your own Service User Number to specify a payment reference for Bacs payments.
- `retry_if_possible`  
On failure, automatically retry payments using [intelligent retries](#). Default is `false`. **Important:** To be able to use intelligent retries, Success+ needs to be enabled in [GoCardless dashboard](#).

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
PUT https://api.gocardless.com/subscriptions/SU123 HTTP/1.1
```

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

```
{
  "subscriptions": {
    "name": "New name",
    "metadata": {
      "order_no": "ABCD4321"
    }
  }
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "subscriptions": {
    "id": "SB123",
    "created_at": "2014-10-20T17:01:06.000Z",
    "amount": 2500,
    "currency": "GBP",
    "status": "active",
    "name": "New name",
    "start_date": "2014-11-03",
    "end_date": null,
    "interval": 1,
    "interval_unit": "monthly",
    "day_of_month": 1,
    "month": null,
    "payment_reference": null,
    "earliest_charge_date_after_resume": null,
    "parent_plan_paused": false,
    "upcoming_payments": [
      { "charge_date": "2014-11-03", "amount": 2500 },
      { "charge_date": "2014-12-01", "amount": 2500 },
      { "charge_date": "2015-01-02", "amount": 2500 },
      { "charge_date": "2015-02-02", "amount": 2500 },
      { "charge_date": "2015-03-02", "amount": 2500 },
      { "charge_date": "2015-04-01", "amount": 2500 },
      { "charge_date": "2015-05-01", "amount": 2500 },
      { "charge_date": "2015-06-01", "amount": 2500 },
      { "charge_date": "2015-07-01", "amount": 2500 },
      { "charge_date": "2015-08-03", "amount": 2500 }
    ],
    "metadata": {
      "order_no": "ABCD4321"
    },
    "links": {
      "mandate": "MA123"
    },
    "retry_if_possible": false
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardlessPro\Environment::SANDBOX;
]);
$client->subscriptions()->update(
  'params' => ['name' => "New name",
  ];
```

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
        "metadata" => ["order_no" => "ABCD4321"]]
]);
Python
```

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.subscriptions.update("SB123", params={
    "name": "New name"
})
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.subscriptions.update(
  "SB123",
  params: {
    metadata: { order_no: "ABCD4321" }
  }
)
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.subscriptions().update("SB123")
    .withName("New name")
    .execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const subscription = await client.subscriptions.update(
  "SB123",
  {
    amount: "42",
    name: "New Name"
});

```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var subscriptionRequest = new GoCardless.Services.SubscriptionUpdateRequest()
{
    Metadata = new Dictionary<string, string>()
    {
        {"custom_reference", "ref_09011991"}
    }
};

var subscriptionResponse = await gocardless.Subscriptions.UpdateAsync("SB0123", subscriptionRequest);
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    subscriptionUpdateParams := &gocardless.SubscriptionUpdateParams{
        Amount: 4200,
        Name: "New Name",
    }
    subscription, err := client.SubscriptionUpdate("SB0123", subscriptionUpdateParams)
}
```

## Pause a subscription

Pause a subscription object. No parameters required.

This can only be used when a subscription has been created without an end date (created without count or

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

When `pause_cycles` is omitted the subscription is paused until the [resume endpoint](#) is called. If the subscription is collecting a fixed number of payments, `end_date` will be set to null. When paused indefinitely, `upcoming_payments` will be empty.

When `pause_cycles` is provided the subscription will be paused for the number of cycles requested. If the subscription is collecting a fixed number of payments, `end_date` will be set to a new value. When paused for a number of cycles, `upcoming_payments` will still contain the upcoming charge dates.

This fails with:

- `forbidden` if the subscription was created by an app and you are not authenticated as that app, or if the subscription was not created by an app and you are authenticated as an app
- `validation_failed` if invalid data is provided when attempting to pause a subscription.
- `subscription_paused_cannot_update_cycles` if the subscription is already paused for a number of cycles and the request provides a value for `pause_cycle`.
- `subscription_CANNOT_be_paused` if the subscription cannot be paused.
- `subscription_already_ended` if the subscription has taken all payments.
- `pause_cycles_must_be_greater_than_or_equal_to` if the provided value for `pause_cycles` cannot be satisfied.

Relative endpoint: POST /subscriptions/SB123/actions/pause

**Warning:** Please note that any payments that have already been created from this subscription will still be collected unless you manually cancel them.

Parameters

#### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

#### pause\_cycles

The number of cycles to pause a subscription for. A cycle is one duration of `interval` and `interval_unit`. This should be a non zero positive value. For AUD subscriptions with `interval_unit`: weekly the minimum value varies between 3 & 4 because of the [mandatory minimum waiting period](#). For NZD subscriptions with `interval_unit`: weekly the minimum value is 2 because of the [mandatory minimum waiting period](#).

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/subscriptions/SU123/actions/pause HTTP/1.1
Content-Type: application/json
{
  "data": {
    "metadata": {}
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "subscriptions": {
    "id": "SB123",
    "created_at": "2014-10-20T17:01:06.000Z",
    "amount": 2500,
    "currency": "GBP",
    "status": "paused",
    "name": "Subscription name",
    "start_date": "2014-11-03",
    "end_date": null,
    "interval": 1,
    "interval_unit": "monthly",
    "day_of_month": 1,
    "month": null,
    "payment_reference": null,
    "earliest_charge_date_after_resume": "2014-10-23",
    "parent_plan_paused": false,
    "upcoming_payments": [
    ],
    "metadata": {
      "order_no": "ABCD4321"
    },
    "links": {
      "mandate": "MA123"
    },
    "retry_if_possible": false
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->subscriptions()->paus
```

Python

```
import gocardless_pro
client = gocardless_pro.Client
```

```
client.subscriptions.pause("SB123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

@client.subscriptions.pause("SB123")
Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.subscriptions().pause("SB123").execute();

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const subscriptionResponse = await client.subscriptions.pause("SB123");

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var subscriptionResponse = await gocardless.Subscriptions.PauseAsync("SB0123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    subscriptionPauseParams := gocardless.SubscriptionPauseParams{}
    subscription, err := client.Subscriptions.Pause(context, "SB123", subscriptionPauseParams)
}

}

```

## Resume a subscription

Resume a subscription object. Payments will start to be created again based on the subscriptions recurrence rules. The `charge_date` on the next payment will be the same as the subscriptions `earliest_charge_date_after_resume`

This fails with:

- `forbidden` if the subscription was created by an app and you are not authenticated as that app, or if the subscription was not created by an app and you are authenticated as an app
- `validation_failed` if invalid data is provided when attempting to resume a subscription.
- `subscription_not_paused` if the subscription is not paused.

Relative endpoint: POST /subscriptions/SB123/actions/resume

**Warning:** There is a mandatory minimum 14 business day wait between resuming an AUD subscription and its next charge date. There is a mandatory minimum 10 calendar day wait between resuming an NZD subscription and its next charge date.

Parameters

**metadata**  
Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com

Content-Type: application/json

```
{
  "data": {
    "metadata": {}
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "subscriptions": {
    "id": "SB123",
    "created_at": "2014-10-20T",
    "amount": 2500,
    "currency": "GBP",
    "status": "active",
    "name": "Subscription name",
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "start_date": "2014-11-03",
    "end_date": null,
    "interval": 1,
    "interval_unit": "monthly",
    "day_of_month": 1,
    "month": null,
    "payment_reference": null,
    "earliest_charge_date_after_resume": null,
    "parent_plan_paused": false,
    "upcoming_payments": [
        { "charge_date": "2014-11-03", "amount": 2500 },
        { "charge_date": "2014-12-01", "amount": 2500 },
        { "charge_date": "2015-01-02", "amount": 2500 },
        { "charge_date": "2015-02-02", "amount": 2500 },
        { "charge_date": "2015-03-02", "amount": 2500 },
        { "charge_date": "2015-04-01", "amount": 2500 },
        { "charge_date": "2015-05-01", "amount": 2500 },
        { "charge_date": "2015-06-01", "amount": 2500 },
        { "charge_date": "2015-07-01", "amount": 2500 },
        { "charge_date": "2015-08-03", "amount": 2500 }
    ],
    "metadata": {
        "order_no": "ABCD4321"
    },
    "links": {
        "mandate": "MA123"
    },
    "retry_if_possible": false
}
}

```

PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->subscriptions()->resume("SB123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.subscriptions.resume("SB123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.subscriptions.resume("SB123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.subscriptions().resume("SB123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const subscriptionResponse = await client.subscriptions.resume("SB123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var subscriptionResponse = await gocardless.Subscriptions.ResumeAsync("SB0123");
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_
    config, err := gocardless.New
    if err != nil {
        fmt.Printf("got err in ini
        return
    }
    client, err := gocardless.New
    if err != nil {
        fmt.Println("error in init
        return
    }
    subscriptionResumeParams := &
    subscription, err := client.Subscriptions.Resume(context, "SB123", subscriptionResumeParams)
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

}

## Cancel a subscription

Immediately cancels a subscription; no more payments will be created under it. Any metadata supplied to this endpoint will be stored on the payment cancellation event it causes.

This will fail with a cancellation\_failed error if the subscription is already cancelled or finished.

Relative endpoint: POST /subscriptions/SB123/actions/cancel

**Warning:** Please note that this will not cancel any payments that have already been created from this subscription — these payments will still be collected unless you also manually cancel them.

Parameters

### metadata

Key-value store of custom data. Up to 3 keys are permitted, with key names up to 50 characters and values up to 500 characters.

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/subscriptions/SU123/actions/cancel HTTP/1.1
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "subscriptions": {
    "id": "SB123",
    "created_at": "2014-10-20T17:01:06.000Z",
    "amount": 2500,
    "currency": "GBP",
    "status": "cancelled",
    "name": "Monthly Magazine",
    "start_date": "2014-11-03",
    "end_date": null,
    "interval": 1,
    "interval_unit": "monthly",
    "day_of_month": 1,
    "month": null,
    "payment_reference": null,
    "earliest_charge_date_after_resume": null,
    "parent_plan_paused": false,
    "upcoming_payments": [],
    "metadata": {
      "order_no": "ABCD1234"
    },
    "links": {
      "mandate": "MA123"
    },
    "retry_if_possible": false
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->subscriptions()->cancel("SB123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
```

```
client.subscriptions.cancel("SB123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

```
@client.subscriptions.cancel("SB123")
```

Java

```
import static com.gocardless.GoCardlessClient.*;
String accessToken = "your_access_token";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
client.subscriptions().cancel()
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');
```

```
const subscriptions = await client.subscriptions().cancel()
```

.NET

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var subscriptionResponse = await gocardless.Subscriptions.CancelAsync("SB0123");
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    subscriptionCancelParams := gocardless.SubscriptionCancelParams{}
    subscription, err := client.Subscriptions.Cancel(context, "SB123", subscriptionCancelParams)
}

```

## Tax Rates

Tax rates from tax authority.

We also maintain a [static list of the tax rates for each jurisdiction](#).

### Properties

<code>id</code>	The unique identifier created by the jurisdiction, tax type and version
<code>end_date</code>	Date at which GoCardless stopped applying the tax rate for the jurisdiction.
<code>jurisdiction</code>	The jurisdiction this tax rate applies to
<code>percentage</code>	The percentage of tax that is applied onto of GoCardless fees
<code>start_date</code>	Date at which GoCardless started applying the tax rate in the jurisdiction.
<code>type</code>	The type of tax applied by this rate

### List tax rates

Returns a [cursor-paginated](#) list of all tax rates.

Relative endpoint: GET /tax\_rates

### Parameters

<code>after</code>	Cursor pointing to the start of the desired set.
<code>before</code>	Cursor pointing to the end of the desired set.
<code>jurisdiction</code>	The jurisdiction this tax rate applies to
<code>limit</code>	Number of records to return.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET [https://api.gocardless.com/tax\\_rates?jurisdiction=GB](https://api.gocardless.com/tax_rates?jurisdiction=GB) HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "tax_rates": [
        {
            "id": "GB_VAT_1",
            "jurisdiction": "G",
            "type": "VAT",
            "percentage": "20.0",
            "start_date": "2021-01-01",
            "end_date": null,
        }
    ],
    "meta": {
        "cursors": {
            "after": null,
            "before": null
        },
        "limit": 50
    }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## PHP

```
$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->taxRates()->list([
    'params' => ["jurisdiction" => "GB"]
]);
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.tax_rates.list(params={"jurisdiction": "GB"}).records
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.tax_rates.list(params: { jurisdiction: "GB" })
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.taxRates().all().withJurisdiction("GB").execute()
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const taxRates = await client.taxRates.list();

// List the Tax rate for a jurisdiction.
const taxRates = await client.taxRates.list({ jurisdiction: 'GB' });
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var request = new GoCardless.Services.TaxRateListRequest()
{
    Jurisdiction = "GB",
};

var response = gocardless.TaxRates.All(request);
foreach (GoCardless.Resources.TaxRate rate in response)
{
    Console.WriteLine(rate.Rate);
}
```

## Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    taxRateListParams := gocardless.TaxRateListParams{
        Jurisdiction: "GB",
    }
    taxRateListResult, err := cl.
```

**Get a single tax rate**

Retrieves the details of a tax rate.

Relative endpoint: GET /tax\_rate

HTTP PHP Python Ruby Java JavaScript .NET Go

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## HTTP

```
GET https://api.gocardless.com/tax_rates/GB_VAT_1 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "tax_rates": [
    {
      "id": "GB_VAT_1",
      "jurisdiction": "GB",
      "type": "VAT",
      "percentage": "20.0",
      "start_date": "2020-02-24",
      "end_date": null
    }
  ]
}
```

## PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->taxRates()->get("GB_VAT_1");
```

## Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.tax_rates.get("GB_VAT_1")
```

## Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.tax_rates.get("GB_VAT_1")
```

## Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

TaxRate taxRate = client.taxRates().get("GB_VAT_1").execute();
```

## JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const taxRate = await client.taxRate.find("GB_VAT_1");
```

## .NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var taxRateResponse = await gocardless.TaxRate.GetAsync("GB_VAT_1");
GoCardless.Resources.TaxRate taxRate = taxRateResponse.TaxRate;
```

## Go

```
package main

import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
  accessToken := "your_access_token_here"
  config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
  if err != nil {
    fmt.Printf("got err in initialising config: %s", err.Error())
    return
  }
  client, err := gocardless.New(config)
  if err != nil {
    fmt.Println("error in initialising client: %s", err.Error())
    return
  }
  taxRate, err := client.TaxRa
}
```

**Transferred Mandate**

Mandates that have been transferred from another customer account.

## Properties

`encrypted_customer_bank_details`

Encrypted customer bank account details, containing: `iban`, `account_holder_name`, `swift_bank_code`, `swift_branch_code`, `swift_account_number`

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

`encrypted_decryption_key`  
Random AES-256 key used to encrypt bank account details, itself encrypted with your public key.

`public_key_id`  
The ID of an RSA-2048 public key, from your JWKS, used to encrypt the AES key.

`links[customer_bank_account]`  
The ID of the updated [customer\\_bank\\_account](#)

`links[mandate]`  
The ID of the transferred mandate

## Get updated customer bank details

Returns new customer bank details for a mandate that's been recently transferred

Relative endpoint: GET /transferred\_mandates/MD123

**Note:** See our [Security Requirements](#) before using this feature

**Restricted:** This endpoint is restricted to organisations with the Transfer Bank Accounts upgrade

Parameters



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/transferred_mandates/MD123 HTTP/1.1
Content-Type: application/json
```

```
HTTP/1.1 200 (OK)
Location: /transferred_mandates/MD123
Content-Type: application/json
{
  "transferred_mandates": {
    "encrypted_decryption_key": "bXktZW5jcnlvcHR1ZC1hZXMa2V5",
    "public_key_id": "9770a024c90fb646e48c952ec5d4f53586e62e8154048e6b96dd9f74f164a472",
    "encrypted_customer_bank_details": "OTc3MGewMjRjOTBmYjY0NmU0GM5NTJ1YzVkJNGY1MzU4NmU2MmU4MTU0MDQ4ZTzi0TZkZD1mNzRmMTY0YTQ3Mg==",
    "links": {
      "customer_bank_account": "BA123",
      "mandate": "MD123"
    }
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->transferred_mandates()->get("MD123");
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.transferred_mandates.get("MD123")
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.transferred_mandates.get("MD123")
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

client.transferredMandates().get("MD123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your')
```

```
const transferredMandate = await
```

.NET

```
String accessToken = "your_ac
GoCardlessClient gocardless =
var response = await gocardles
Go
```

```
package main
```

```
import (
  gocardless "github.com/gocardless/gocardless-pro-go/v2"
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

)
func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    transferredMandate, err := client.TransferredMandates.Get(context, "MD123")
}

```

## Verification Details

Verification details represent any information needed by GoCardless to verify a creditor.

**Restricted:** These endpoints are restricted to customers who want to collect their merchant's verification details and pass them to GoCardless via our API. Please [get in touch](#) if you wish to enable this feature on your account.

### Properties

`address_line1`  
The first line of the company's address.

`address_line2`  
The second line of the company's address.

`address_line3`  
The third line of the company's address.

`city`  
The city of the company's address.

`company_number`  
The company's registration number.

`description`  
A summary describing what the company does.

`directors`  
The company's directors.

Each instance will contain these properties:

- `city`: The city of the person's address.
- `country_code`: [ISO 3166-1 alpha-2 code](#).
- `date_of_birth`: The person's date of birth.
- `family_name`: The person's family name.
- `given_name`: The person's given name.
- `postal_code`: The person's postal code.
- `street`: The street of the person's address.

`name`  
The company's legal name.

`postal_code`  
The company's postal code.

`links[creditor]`  
ID of the [creditor](#)

## Create a verification detail

Creates a new verification detail

Relative endpoint: POST /verification\_details

**Warning:** This endpoint only supports UK based limited companies. Therefore the creditor's `creditor_type` must be 'company' and `country_code` must be 'GB'.

### Parameters

`address_line1`  
*required* The first line of the company's address.

`city`  
*required* The city of the company's address.

`company_number`  
*required* The company's registration number.

`description`  
*required* A summary describing what the company does.

`directors`  
*required* The company's directors.

`name`  
*required* The company's legal name.

`postal_code`  
*required* The company's postal code.

`address_line2`  
The second line of the company's address.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

The second line of the company's address.

`address_line3`

The third line of the company's address.

`links[creditor]`

required ID of the associated [creditor](#).

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/verification_details/ HTTP/1.1
Content-Type: application/json
```

```
{
```

```
  "verification_details": {
    "name": "Acme Limited",
    "company_number": "03768189",
    "description": "wine and cheese seller",
    "address_line1": "12 Drury lane",
    "city": "London",
    "postal_code": "B4 7NJ",
    "directors": [
      {
        "date_of_birth": "1986-02-19",
        "given_name": "Gandalf",
        "family_name": "Grey",
        "city": "London",
        "street": "Drury Lane",
        "postal_code": "B4 7NJ",
        "country_code": "GB"
      }
    ],
    "links": {
      "creditor": "CR123"
    }
  }
```

```
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
```

```
  "verification_details": {
    "name": "Acme Limited",
    "company_number": "03768189",
    "description": "wine and cheese seller",
    "address_line1": "12 Drury lane",
    "address_line2": null,
    "address_line3": null,
    "city": "London",
    "region": null,
    "postal_code": "B4 7NJ",
    "country_code": "GB",
    "directors": [
      {
        "given_name": "Gandalf",
        "family_name": "Grey",
        "date_of_birth": "1986-02-19",
        "street": "Drury lane",
        "city": "London",
        "postal_code": "B4 7NJ",
        "country_code": "GB",
      }
    ],
    "links": {
      "creditor": "CR123"
    }
  }
```

```
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$verification_details = $client->verificationDetails()->create([
  'params' => [
    'name' => 'Acme',
    'company_number' => '03768189',
    'address_line1' => '12 Drury lane',
    'city' => 'London',
    'description' => 'wine and cheese seller',
    'postal_code' => 'B4 7NJ',
    'directors' => [
      [
        'given_name' => 'Gandalf',
        'family_name' => 'Grey',
        'city' => 'London',
        'date_of_birth' => '1986-02-19',
        'street' => 'Drury lane',
        'postal_code' => 'B4 7NJ',
        'country_code' => 'GB'
      ]
    ],
    'links' => [
      'creditor' => 'CR123'
    ]
  ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client

client.verification_details.create(
  "name": "Acme",
  "company_number": "03768189",
  "address_line1": "12 Drury lane",
  "city": "London",
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

"description": "wine and cheese seller",
"postal_code": "B4 7NJ",
"directors": [
  {
    "given_name": "Gandalf",
    "family_name": "Grey",
    "city": "London",
    "date_of_birth": "1986-02-19",
    "street": "Drury lane",
    "postal_code": "B4 7NJ",
    "country_code": "GB"
  }
],
"links": {
  "creditor": "CR123"
}
})
Ruby

```

```

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.verification_details.create(params: {
  name: "Acme",
  company_number: "03768189",
  address_line1: "12 Drury lane",
  city: "London",
  description: "wine and cheese seller",
  postal_code: "B4 7NJ",
  directors: [
    {
      given_name: "Gandalf",
      family_name: "Grey",
      city: "London",
      date_of_birth: "1986-02-19",
      street: "Drury lane",
      postal_code: "B4 7NJ",
      country_code: "GB"
    }
  ],
  links: {
    creditor: "CR123",
  },
})

```

## Java

```

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

VerificationDetailService.VerificationDetailCreateRequest.Directors director = new VerificationDetailService.VerificationDetailCreateRequest.Directors

director.withCity("London");
director.withCountryCode("GB");
director.withDateOfBirth("1986-02-19");
director.withGivenName("Gandalf");
director.withFamilyName("Grey");
director.withPostalCode("B4 7NJ");
director.withStreet("Drury Lane");

VerificationDetail verificationDetail = client.verificationDetails().create()
  .withName("Acme")
  .withDescription("wine and cheese seller")
  .withCompanyNumber("03768189")
  .withAddressLine1("12 Drury Lane")
  .withCity("London")
  .withPostalCode("B4 7NJ")
  .withDirectors(director)
  .withLinksCreditor("CR123")
  .execute();

```

## JavaScript

```

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const verificationDetail = await client.verificationDetails.create({
  name: "Acme",
  company_number: "03768189",
  address_line1: "12 Drury lane",
  city: "London",
  description: "wine and cheese seller",
  postal_code: "B4 7NJ",
  directors: [
    {
      given_name: "Gandalf",
      family_name: "Grey",
      city: "London",
      date_of_birth: "1986-02-19",
      street: "Drury lane",
      postal_code: "B4 7NJ",
      country_code: "GB"
    }
  ],
  links: {
    creditor: "CR123"
  }
});

```

## .NET

```

String accessToken = "your_acce
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

var verificationDetailCreateRequest = new VerificationDetailCreateRequest()
{
    Name = "Acme",
    CompanyNumber = "03768189",
    AddressLine1 = "12 Drury lane",
    City = "London",
    Description = "wine and cheese seller",
    PostalCode = "B4 7NJ",
    Directors = new VerificationDetailCreateRequest.VerificationDetailDirectors[]
    {
        new VerificationDetailCreateRequest.VerificationDetailDirectors
        {
            GivenName = "Gandalf",
            FamilyName = "Grey",
            City = "London",
            DateOfBirth = "1986-02-19",
            Street = "Drury lane",
            PostalCode = "B4 7NJ",
            CountryCode = "GB"
        }
    },
    Links = new VerificationDetailCreateRequest.VerificationDetailLinks()
    {
        Creditor = "CR123"
    }
};

var verificationDetailResponse = await client.VerificationDetails.CreateAsync(verificationDetailCreateRequest);

var verificationDetail = verificationDetailResponse.VerificationDetail;
Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    context := context.Background()
    verificationDetailCreateParams := gocardless.VerificationDetailCreateParams{
        Name:          "Acme",
        CompanyNumber: "03768189",
        AddressLine1:   "12 Drury lane",
        City:          "London",
        Description:   "wine and cheese seller",
        PostalCode:    "B4 7NJ",
        Directors: []gocardless.VerificationDetailCreateParamsDirectors{
            {
                GivenName:   "Gandalf",
                FamilyName:  "Grey",
                City:        "London",
                DateOfBirth: "1986-02-19",
                Street:      "Drury Lane",
                PostalCode:  "B4 7NJ",
                CountryCode: "GB",
            },
        },
        Links: gocardless.VerificationDetailCreateParamsLinks{
            Creditor: "CR123",
        },
    }
}

verificationDetail, err := client.VerificationDetails.Create(context, verificationDetailCreateParams)
if err != nil {
    fmt.Printf("error creating verification detail: %s", err.Error())
    return
}
}

```

## List verification details

Returns a list of verification detail

Relative endpoint: GET /verificationdetails

Parameters

**creditor** *required* Unique identifier,

**after** Cursor pointing to the start

**before** Cursor pointing to the end o

**limit** Number of records to return.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
GET https://api.gocardless.com/verification_details?creditor=CR123 HTTP/1.1
```

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
  "verification_details": [
    {
      "name": "Acme Limited",
      "company_number": "03768189",
      "description": "wine and cheese seller",
      "address_line1": "12 Drury lane",
      "address_line2": null,
      "address_line3": null,
      "city": "London",
      "region": null,
      "postal_code": "B4 7NJ",
      "country_code": "GB",
      "directors": [
        {
          "given_name": "Gandalf",
          "family_name": "Grey",
          "date_of_birth": "1986-02-19",
          "street": "Drury lane",
          "city": "London",
          "postal_code": "B4 7NJ",
          "country_code": "GB",
        }
      ],
      "links": {
        "creditor": "CR123"
      }
    }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$response = $client->verificationDetails()->list([
  'params' => [
    'creditor' => 'CR123'
  ]
])->records;
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.verification_details.list(params={
  "creditor": "CR123"
}).records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.verification_details.list(params: {
  creditor: "CR123",
}).records
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

for (VerificationDetail verificationDetail: client.verificationDetails().all().withCreditor("CR123").execute()){
  System.out.println(verificationDetail.getDescription());
}
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const verificationDetailList =
```

.NET

```
String accessToken = "your_acco
GoCardlessClient gocardless = (
```

```
var verificationDetailListRequest
{
  Creditor = "CR123"
};

var verificationDetailListResponse
foreach (var verificationDetail:
{
  Console.WriteLine(verificationDetail);
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    context := context.Background()
    verificationDetailListParams := gocardless.VerificationDetailListParams{
        Creditor: "CR123",
    }
    verificationDetailList, err := client.VerificationDetails.List(context, verificationDetailListParams)
    if err != nil {
        fmt.Printf("error listing verification details: %v", err)
        return
    }

    for _, verificationDetail := range verificationDetailList.VerificationDetails {
        fmt.Println(verificationDetail.Name)
        fmt.Println(verificationDetail.Description)
    }
}

```

## Webhooks

Basic description of a webhook

### Properties

**id**  
Unique identifier, beginning with “WB”.

**created\_at**  
Fixed [timestamp](#), recording when this resource was created.

**is\_test**  
Boolean value showing whether this was a demo webhook for testing

**request\_body**  
The body of the request sent to the webhook URL

**request\_headers**  
The request headers sent with the webhook

**response\_body**  
The body of the response from the webhook URL

**response\_body\_truncated**  
Boolean value indicating the webhook response body was truncated

**response\_code**  
The response code from the webhook request

**response\_headers**  
The headers sent with the response from the webhook URL

**response\_headers\_content\_truncated**  
Boolean indicating the content of response headers was truncated

**response\_headers\_count\_truncated**  
Boolean indicating the number of response headers was truncated

**successful**  
Boolean indicating whether the request was successful or failed

**url**  
URL the webhook was POST-ed to

## List webhooks

Returns a [cursor-paginated](#) list of your webhooks.

Relative endpoint: GET /webhooks

### Parameters

**after**  
Cursor pointing to the start

**before**  
Cursor pointing to the end

**created\_at[gt]**  
Limit to records created after

**created\_at[gte]**  
Limit to records created on or after

**created\_at[lt]**  
Limit to records created before

**created\_at[lte]**  
Limit to records created on or before

**is\_test**

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Show only test/non test webhooks

limit

Number of records to return.

successful

Show only successful/failed webhooks

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
GET https://api.gocardless.com/webhooks/WB123?successful=true HTTP/1.1
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "meta": {
    "cursors": {
      "after": null,
      "before": null
    },
    "limit": 5
  },
  "webhooks": [
    {
      "created_at": "2018-03-12T15:13:37.416Z",
      "id": "WB123",
      "is_test": true,
      "request_body": "{\"events\": [{\"id\": \"EV123\", \"created_at\": \"2018-03-12T15:13:37.158Z\", \"resource_type\": \"payments\", \"action\": \"created\", \"request_headers\": {\"Content-Type\": \"application/json\", \"Origin\": \"https://api.gocardless.com\", \"User-Agent\": \"gocardless-webhook-service/1.1\", \"Webhook-Signature\": \"e4d043149b4cc27435d05ea275a09de2f810e45bed5448fd6a0a742a3846b365\"}, \"response_body\": \"ok\", \"response_body_truncated\": false, \"response_code\": 200, \"response_headers\": {\"content-type\": \"text/html; charset=utf-8\", \"date\": \"Mon, 12 Mar 2018 15:13:37 GMT\"}, \"response_headers_content_truncated\": false, \"response_headers_count_truncated\": false, \"successful\": true, \"url\": \"https://example.com/webhook_handler\"}]}]
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment' => \GoCardlessPro\Environment::SANDBOX
]);

$client->webhooks()->list();
```

```
$client->webhooks()->list([
  'params' => ["successful" => True]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.webhooks.list().records
```

```
client.webhooks.list(params={"successful": True}).records
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.webhooks.list
```

```
@client.webhooks.list(params: { successful: true })
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
for (Webhook webhook : client)
```

```
  System.out.println(webhook.g...
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your...
```

```
const webhooks = await client...
```

```
// List all webhooks past a certain point
const webhooks = await client.w...
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        gt: "2020-01-01T17:01:06.000Z"
    }
});
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var webhookRequest = new GoCardless.Services.WebhookListRequest()
{
    Successful = true
};

var webhookListResponse = gocardless.Webhooks.All(webhookRequest);
foreach (GoCardless.Resources.Webhook webhook in webhookListResponse)
{
    Console.WriteLine(webhook.Id);
}

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    webhookListParams := gocardless.WebhookListParams{
        Createdat: &gocardless.WebhookListParamsCreatedAt{
            Gt: "2020-01-01T17:01:06.000Z",
        },
    }

    webhookListResult, err := client.Webhooks.List(context, webhookListParams)
    for _, webhook := range webhookListResult.Webhooks {
        fmt.Println(webhook.Id)
    }
}
}

```

## Get a single webhook

Retrieves the details of an existing webhook.

Relative endpoint: GET /webhooks/WB123



HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

GET https://api.gocardless.com/webhooks/WB123 HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
{
    "webhooks": [
        {
            "created_at": "2018-03-12T15:13:37.416Z",
            "id": "WB123",
            "is_test": true,
            "request_body": "{\"events\": [{\"id\": \"EV123\", \"created_at\": \"2018-03-12T15:13:37.158Z\", \"resource_type\": \"payments\", \"action\": \"created\"}]}",
            "request_headers": {
                "Content-Type": "application/json",
                "Origin": "https://api.gocardless.com",
                "User-Agent": "gocardless-webhook-service/1.1",
                "Webhook-Signature": "e4d043149b4cc27435d05ea275a09de2f810e45bed5448fd6a0a742a3846b365"
            },
            "response_body": "ok",
            "response_body_truncated": false,
            "response_code": 200,
            "response_headers": {
                "content-type": "text/html; charset=utf-8",
                "date": "Mon, 12 Mar 2019 15:13:27 GMT"
            },
            "response_headers_content": null,
            "response_headers_count": 1,
            "successful": true,
            "url": "https://example.com"
        }
    ]
}
```

PHP

```
$client = new \GoCardlessPro\Client();
$access_token' => 'your_access_token';
'environment' => \GoCardless\Environment::SANDBOX;
]);
$client->webhooks()->get("WB123");
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
```

Ruby

```
@client = GoCardlessPro::Client.new(  
  access_token: "your_access_token",  
  environment: :sandbox  
)
```

`@cli`

```
Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

Webhook webhook = client.webhooks().get("WB123").execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const webhook = await client.webhooks.find("WB123");
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient goCardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var webhookResponse = await goCardless.webhooks.GetAsync("WB123");
GoCardless.Resources.Webhook webhook = webhookResponse.Webhook;
```

Go

```
package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    webhook, err := client.Webhooks.Get(context, "WB123")
}
```

## Retry a webhook

Requests for a previous webhook to be sent again

Relative endpoint: POST /webhooks/WB123/actions/retry

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

POST https://api.gocardless.com/webhooks/WB123/actions/retry HTTP/1.1

HTTP/1.1 200 OK  
Content-Type: application/json

```
  "webhooks": {
    "created_at": "2018-03-12T15:13:37.416Z",
    "id": "WB123",
    "is_test": true,
    "request_body": "{\"even",
    "request_headers": {
        "Content-Type": "appli",
        "Origin": "https://api",
        "User-Agent": "gocardli",
        "Webhook-Signature": "d"
    },
    "response_body": "ok",
    "response_body_truncated": false,
    "response_code": 200,
    "response_headers": {
        "content-type": "text/ln",
        "date": "Mon, 12 Mar 2018 15:13:37 GMT"
    },
    "response_headers_content_truncated": true
}
```

Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

payments\"}, {"action": "created",

```

        "response_headers_count_truncated": false,
        "successful": true,
        "url": "https://example.com/webhook_handler"
    }
}
PHP

$client = new \GoCardlessPro\Client([
    'access_token' => 'your_access_token_here',
    'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
}

$client->webhooks()->retry("WB123");

Python

import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')
client.webhooks.retry("WB123")

Ruby

@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.webhooks.retry("WB123")

Java

import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
    .newBuilder(accessToken)
    .withEnvironment(SANDBOX)
    .build();

client.webhooks().retry("WB123").execute();

JavaScript

const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

const webhookResponse = await client.webhooks.retry("WB123");

.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var webhookResponse = await gocardless.webhooks.RetryAsync("WB123");

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    webhook, err := client.Webhooks.Retry(context, "WB123")
}

```

## Event Actions

All events are associated with a resource. This section goes through each resource in turn, and lists all the possible actions that can relate to that resource. Some actions have different variations - these are listed in the relevant tables below each action.

### Billing Request

This is a list of all the different variations of the Billing Request event:

#### created

This billing request has been created.

#### Origin Cause

Origin	Cause
gocardless	billing_request_create
api	billing_request_create
payer	billing_request_created

This billing request has been created.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**flow\_created**

A billing request flow has been created against this billing request.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	billing_request_flow_created	A billing request flow has been created against this billing request.
api	billing_request_flow_created	A billing request flow has been created against this billing request.
payer	billing_request_flow_created	A billing request flow has been created against this billing request.

**flow\_visited**

The billing request flow has been visited.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_flow_visited	The billing request flow has been visited.

**flow\_exited**

The billing request flow has been exited by the payer.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_flow_exited	The billing request flow has been exited by the payer.

**collect\_customer\_details**

Customer details have been collected for this billing request.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
api	billing_request_collect_customer_details	Customer details have been collected for this billing request.
payer	billing_request_collect_customer_details	Customer details have been collected for this billing request.

**select\_institution**

Institution details have been collected for this billing request.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_select_institution	Institution details have been collected for this billing request.

**collect\_bank\_account**

Bank account details have been collected for this billing request.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
api	billing_request_collect_bank_account	Bank account details have been collected for this billing request.
payer	billing_request_collect_bank_account	Bank account details have been collected for this billing request.

**payer\_details\_confirmed**

Payer has confirmed all their details for this billing request.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
api	billing_request_payer_details_confirmed	Payer has confirmed all their details for this billing request.
payer	billing_request_payer_details_confirmed	Payer has confirmed all their details for this billing request.

**bank\_authorisation\_visited**

A bank authorisation link for this billing request has been visited.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_bank_authorisation_visited	A bank authorisation link for this billing request has been visited.

**bank\_authorisationauthorised**

A bank authorisation for this billing request has been authorised by the payer.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_bank_authorisationauthorised	A bank authorisation for this billing request has been authorised by the payer.
gocardless	billing_request_bank_authorisationauthorised	A bank authorisation for this billing request has been authorised by the payer.
<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_bank_authorisationauthorised	A bank authorisation for this billing request has been authorised by the payer.
gocardless	billing_request_bank_authorisationauthorised	A bank authorisation for this billing request has been authorised by the payer.

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

 Consultez notre politique de confidentialité

**bank\_authorisation\_expired**

A bank authorisation for this billing request has expired.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_bank_authorisation_expired	A bank authorisation for this billing request has expired.

#### **bank\_authorisation\_failed**

A bank authorisation for this billing request has failed.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_bank_authorisation_failed	A bank authorisation for this billing request has failed.
gocardless	billing_request_bank_authorisation_failed	A bank authorisation for this billing request has failed.
gocardless	insufficient_funds	A bank authorisation for this billing request has failed due to insufficient funds.

#### **fulfilled**

This billing request has been fulfilled, and the resources have been created.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	billing_request_fulfilled	This billing request has been fulfilled, and the resources have been created.
api	billing_request_fulfilled	This billing request has been fulfilled, and the resources have been created.

#### **cancelled**

This billing request has been cancelled, none of the resources have been created.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	billing_request_cancelled	This billing request has been cancelled, none of the resources have been created.
api	billing_request_cancelled	This billing request has been cancelled, none of the resources have been created.

#### **choose\_currency**

Currency details have been collected for this billing request.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
api	billing_request_choose_currency	Currency details have been collected for this billing request.
payer	billing_request_choose_currency	Currency details have been collected for this billing request.

#### **collect\_amount**

Amount has been collected for this billing request.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	billing_request_collect_amount	Amount has been collected for this billing request.

#### **payer\_geo\_blocked**

Payer blocked for 24 hours for attempting to pay from an unsupported location.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
payer	payer_geo_blocked	Payer blocked for 24 hours for attempting to pay from an unsupported location.

#### **failed**

This billing request has failed.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	billing_request_failed	This billing request has failed.
api	billing_request_failed	This billing request has failed.

HTTP  
HTTP

```
GET https://api.gocardless.com/events/EV123 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
Content-Type: application/json
{
```

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "billing_request": "BRQ123"
    }
}
}
```

## Creditor

This is a list of all the different values for `action`

### **updated**

Something has changed about this creditor. The property that has changed will be included in the event details. Currently, this webhook is sent for `logo_url`, `verification_status`, `mandate_imports_enabled`, `custom_payment_pages_enabled` and `merchant_responsible_for_notifications`.

Origin	Cause	Description
--------	-------	-------------

gocardless creditor\_updated This creditor has been updated.

### **new\_payout\_currency\_added**

This creditor has added a new payout currency.

Origin	Cause	Description
--------	-------	-------------

gocardless new\_payout\_currency\_added This creditor has added a new payout currency.

### **account\_auto\_frozen**

This creditor account has been automatically frozen and had restrictions applied.

Origin	Cause	Description
--------	-------	-------------

gocardless account\_auto\_frozen This creditor account has been automatically frozen and had restrictions applied.

### **account\_auto\_frozen\_reverted**

This creditor accounts restrictions have been removed.

Origin	Cause	Description
--------	-------	-------------

gocardless account\_auto\_frozen\_reverted The restrictions on this creditor account have been removed.

### **bounced\_payout**

A payout for this creditor has failed. Please contact your bank for more information or retry the payout.

Origin	Cause	Description
--------	-------	-------------

gocardless bounced\_payout A payout for this creditor has failed. Please retry the payout or contact your bank for more information.



HTTP

HTTP

```
GET https://api.gocardless.com/events/EV123 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
```

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

```
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2018-04-08T17:01:06.000Z",
      "resource_type": "creditors",
      "action": "updated",
      "links": [
        {
          "creditor": "CR123"
        }
      ],
      "details": [
        {
          "origin": "gocardless",
          "cause": "creditor_updated",
          "description": "This creditor has been updated.",
          "property": "fx_payout_currency"
        }
      ],
      "metadata": {}
    }
  ]
}
```

## Export

This is a list of all the different va

### **started**

Export started

Origin	Cause	Description
--------	-------	-------------

gocardless export\_started Export

### **completed**

Export completed

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Description
gocardless	export_completed	Export completed

**failed**

Export failed

Origin	Cause	Description
gocardless	export_failed	Export failed



HTTP

HTTP

```
GET https://api.gocardless.com/events/EV123 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
```

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

```
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2024-07-10T00:00:00.000Z",
      "resource_type": "exports",
      "action": "started",
      "links": [
        {
          "export": "EX0000116MFWM"
        }
      ],
      "details": [
        {
          "origin": "gocardless",
          "cause": "export_started",
          "description": "Export started.",
          "type": "payout_transactions_reconciliation",
          "currency": "GBP"
        }
      ],
      "metadata": {}
    }
  ]
}
```

## Instalment Schedule

This is a list of all the different values for `action`

**created**

The instalment schedule has been created.

Origin	Cause	Description
api	instalment_schedule_created	Instalment schedule has been created via the API

**creation\_failed**

The instalment schedule failed to create due to validation errors when creating the payments. The errors will be included in the event payload.

Origin	Cause	Description
api	instalment_schedule_creation_failed	Instalment schedule failed to be created

**cancelled**

The instalment schedule has been cancelled. Any pending payments have also been cancelled.

Origin	Cause	Description
api	instalment_schedule_cancelled	Instalment schedule has been cancelled
gocardless	mandate_cancelled	Instalment schedule has been cancelled
gocardless	mandate_failed	Instalment schedule has been cancelled
gocardless	mandate_suspended_by_payer	Instalment schedule has been cancelled
gocardless	mandate_expired	Instalment schedule has been cancelled

**errored**

One or more instalments in this instalment schedule failed to collect successfully.

Origin	Cause	Description
gocardless	instalment_schedule_error	
gocardless	instalment_schedule_error	

**resumed**

The instalment schedule has been resumed.

Origin	Cause	Description
gocardless	instalment_schedule_resumed	

**completed**

This instalment schedule has concluded. No further instalments will be collected.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Description
gocardless	instalment_schedule_completed	Instalment schedule has completed
HTTP		
HTTP		
GET https://api.gocardless.com/events/EV123		HTTP/1.1 200 (OK)
		Content-Type: application/json
		{
		"events": {
		"id": "EV123",
		"created_at": "2014-04-08T17:01:06.000Z",
		"resource_type": "instalment_schedules",
		"action": "payment_created",
		"links": {
		"instalment_schedule": "IS123",
		"payment": "PM123"
		},
		"details": {
		"origin": "api",
		"cause": "payment_created",
		"description": "Payment created by an instalment schedule."
		},
		"metadata": {},
		"resource_metadata": {
		"order_dispatch_date": "2014-05-22"
		}
		}
		}

## Mandate

This is a list of all the different values for action

### created

The mandate has been created.

Origin	Cause	Description
api	mandate_created	Mandate created via the API.
gocardless	mandate_created	Mandate created by a bulk change

### customer\_approval\_granted

The mandate required additional approval from the customer (e.g. permission from a second signatory), and that approval has been granted.

Origin	Cause	Description
customer	customer_approval_granted	The customer has granted approval for this mandate

### customer\_approval\_skipped

The mandate originally was believed to require additional approval from the customer (e.g. permission from a second signatory), but approval has been skipped (for example because the mandate was erroneously marked as needing a second signature).

Origin	Cause	Description
customer	customer_approval_skipped	The customer has skipped approval for this mandate

### active

The mandate has been successfully set up by the customer's bank.

Origin	Cause	Description
gocardless	mandate_activated	The time window after submission for the banks to refuse a mandate has ended without any errors being received, so this mandate is now active.
bank	mandate_activated	The customer's bank has confirmed that this mandate has been activated.

### cancelled

The mandate has been cancelled, either by the customer through their bank or this API, or automatically when their bank account is closed.

Origin	Cause	Nous utilisons des cookies	Description
bank	bank_account_closed	Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.	has been closed as the customer is deceased.
bank	bank_account_closed		unt for this mandate has been closed.
bank	mandate_cancelled		as cancelled at a bank branch.
bank	authorisation_disputed		utes having authorised you to set mandate with them.
bank	invalid_bank_details	 Consultez notre politique de confidentialité	nk account does not exist or was closed.

Origin	Cause	Scheme	Reason code	Description
bank	direct_debit_not_enabled	ach	ACH_RETURN-R16ACH_RETURN-R34	The bank account does not support Direct Debit.
bank	refer_to_payer	ach	ACH_RETURN-R05ACH_RETURN-R06ACH_RETURN-R08ACH_RETURN-R16ACH_RETURN-R20ACH_RETURN-R29ACH_RETURN-R26	This mandate has been cancelled because a payment under it failed.
bank	return_on_odfi_request	ach	ACH_RETURN-R06	mandate_cancelled_because_payment_charged_back
api	mandate_cancelled	-	-	The mandate was cancelled at your request.
api	bank_account_closed	-	-	The customer's account was disabled at your request.
bank	mandate_cancelled	bacs	ADDACS-0ADDACS-1	The mandate was cancelled at a bank branch.
bank	bank_account_closed	bacs	ADDACS-2	This bank account has been closed as the customer is deceased.
bank	bank_account_transferred	bacs	ADDACS-3ADDACS-CADDACS-E	The customer's bank account was transferred to a different bank or building society.
bank	bank_account_closed	bacs	ADDACS-B	The customer's account was closed at their bank.
bank	authorisation_disputed	bacs	ADDACS-D	The customer has disputed the amount of notice specified on the mandate via their bank.
bank	mandate_cancelled	bacs	ARUDD-1	The mandate was cancelled at a bank branch.
bank	bank_account_closed	bacs	ARUDD-2	This mandate has been cancelled, because a payment against it indicated that the customer is deceased.
bank	bank_account_closed	bacs	ARUDD-B	The bank account for this mandate has been closed.
bank	refer_to_payer	bacs	ARUDD-6	This mandate has been cancelled because a payment under it failed.
bank	authorisation_disputed	bacs	DDICA-4	The customer claims that they asked you to cancel their mandate before you took the payment.
bank	authorisation_disputed	bacs	DDICA-5DDICA-6DDICA-8	The customer disputes having authorised you to set up a mandate with them.
bank	invalid_bank_details	sepa_core	AC01BE06	The bank account specified does not exist. Any subscriptions and pending payments will also be cancelled.
bank	bank_account_closed	sepa_core	AC04	The bank account for this mandate has been closed. Any subscriptions and pending payment will also be cancelled.
bank	direct_debit_not_enabled	sepa_core	AG01AC06	The bank account for this mandate does not support SEPA direct debit. Any subscriptions and pending payments will also be cancelled.
bank	account_blocked_for_any_financial_transaction	sepa_core	AC06	The bank account for this mandate was blocked. Any subscriptions and pending payments will also be cancelled.
bank	mandate_cancelled	sepa_core	MD01	A payment under this mandate failed, indicating that the mandate has been cancelled at the customer's bank. Any subscriptions and pending payments will also be cancelled.
bank	bank_account_closed	sepa_core	MD07	This mandate has been cancelled because the customer is deceased.
bank	mandate_cancelled	betalingsservice	MANDATE_INFORMATION-0232MANDATE_INFORMATION-0233MANDATE_INFORMATION-0234	This mandate was cancelled by the customer or their bank.
bank	mandate_cancelled	betalingsservice	INFORMATION_LIST-0257	This mandate was cancelled by the customer or their bank.
bank	direct_debit_not_enabled	becs	2	The bank account does not support Direct Debit.
bank	payment_stopped	becs	2	The payment was stopped by the payer or their bank.
bank	other	becs	2	This mandate has been cancelled because a payment under it failed.
bank	bank_account_closed	becs	3	The bank account for this mandate has been closed.
bank	bank_account_closed	becs	4	This bank account has been closed as the customer is deceased.
bank	authorisation_disputed	becs	UDUNAUT	The customer disputes having authorised you to set up a mandate with them.
bank	mandate_cancelled			as cancelled by the customer or their bank.
bank	invalid_bank_details			ount specified does not exist.
bank	authorisation_disputed			utes having authorised you to set mandate with them.
bank	bank_account_closed			t account was closed at their bank.
bank	bank_account_closed			t has been closed as the customer is deceased.
bank	invalid_bank_details			nt specified does not exist. Any pending payments will also be cancelled.
bank	refer_to_payer			internal error processing this mandate.

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	mandate_cancelled	pay_to	MD16MS02	This mandate was cancelled by the customer or their bank.
bank	mandate_cancelled	pay_to	MD17	The mandate was cancelled at your request.
bank	mandate_cancelled	pay_to	MD21	The mandate has been cancelled due to fraud.
bank	mandate_cancelled	pay_to	MSUCMCFCMCOC	The mandate has been suspended.
bank	refer_to_payer	pay_to	UNKN	This mandate was cancelled by the customer or their bank.
bank	invalid_bank_details	pay_to	AC02	This mandate was cancelled due to a Notification of Change indicating the customer's account number or branch number was incorrect, please contact the customer.
bank	other	pay_to	AC06	The account associated with the customer's mandate is blocked.
bank	invalid_bank_details	pay_to	AC13	This mandate was cancelled due to a Notification of Change indicating the customer's selected account type was incorrect, please contact the customer.
bank	bank_account_closed	pay_to	AC04AC05	The customer's account was closed at their bank.
bank	bank_account_closed	pay_to	MD07	This mandate has been cancelled because the customer is deceased.
bank	other	pay_to	SL11SL12	The customer's account is not permitted to send funds to the beneficiary.
bank	bank_account_closed	becs_nz	DISHONOUR-AC04	The customer's account was closed at their bank.
bank	invalid_bank_details	becs_nz	DISHONOUR-AC05NEGATIVE_ACKNOWLEDGEMENT-AC05	The bank account specified does not exist. Any subscriptions and pending payments will also be cancelled.
bank	bank_account_closed	pad	PAYMENT_STATUS_REPORT-905	The customer's account was closed at their bank.
bank	bank_account_closed	pad	PAYMENT_STATUS_REPORT-910	This bank account has been closed as the customer is deceased.
bank	authorisation_disputed	pad	PAYMENT_STATUS_REPORT-915	The customer disputes having authorised you to set up a mandate with them.
bank	mandate_cancelled	pad	PAYMENT_STATUS_REPORT-917PAYMENT_STATUS_REPORT-920	This mandate was cancelled by the customer or their bank.
bank	invalid_bank_details	pad	PAYMENT_STATUS_REPORT-0518PAYMENT_STATUS_REPORT-0567	The specified bank account does not exist or was closed.
bank	mandate_cancelled	faster_payments	PAYER_MANDATE_CANCELLED	This mandate was cancelled by the customer or their bank.
bank	mandate_cancelled	faster_payments	MANDATE_CANCELLED_BY_BANK	This mandate was cancelled by the bank.
bank	mandate_cancelled	faster_payments	FAILED_ELIGIBILITY_CHECK	This mandate was cancelled because the bank account has transitioned to an inactive state.
gocardless	mandate_cancelled	faster_payments	MERCHANT_MANDATE_CANCELLED	The mandate was cancelled at your request.
gocardless	initial_one_off_payment_failed	-	-	This mandate has been cancelled because the initial faster payment failed

## failed

The mandate could not be set up, generally because the specified bank account does not accept Direct Debit payments or is closed.

Origin	Cause	Scheme	Reason code	Description
bank	bank_account_closed	ach	ACH_RETURN-R14ACH_RETURN-R15	This bank account has been closed as the customer is deceased.
bank	refer_to_payer	ach	ACH_RETURN-C01ACH_RETURN-C02ACH_RETURN-C03ACH_RETURN-C06ACH_RETURN-C07NOTIFICATION_OF_CHANGE-C01NOTIFICATION_OF_CHANGE-C02NOTIFICATION_OF_CHANGE-C03NOTIFICATION_OF_CHANGE-C06NOTIFICATION_OF_CHANGE-C07NOTIFICATION_OF_CHANGE-C08NOTIFICATION_OF_CHANGE-C09	This mandate was cancelled due to a Notification of Change indicating the customer's account number or branch number was incorrect, please contact the customer.
bank	refer_to_payer	ach	ACH_RETURN-C05NOTIFICATION_OF_CHANGE-C05	This mandate was cancelled due to a Notification of Change indicating the customer's selected account type was incorrect, please contact the customer.
bank	bank_account_closed	ach	ACH_RETURN-R02ACH_RETURN-R12	The bank account for this mandate has been closed.
bank	mandate_cancelled	ach		The mandate was already cancelled.
bank	authorisation_disputed	ach		The mandate was already cancelled.
bank	invalid_bank_details	ach		The specified bank account does not exist or was closed.
bank	direct_debit_not_enabled	ach		The bank account does not support Direct Debit.
bank	refer_to_payer	ach		This mandate has been cancelled because a payment under it failed.
bank	other	ach		This mandate has been cancelled because a payment under it failed.

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

 Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	return_on_odfi_request	ach	ACH_RETURN-R06	The mandate has been cancelled because a payment under it was charged back.
bank	authorisation_disputed	ach	ACH_RETURN-R05ACH_RETURN-R08ACH_RETURN-R11	The customer disputes having authorised you to set up a mandate with them.
bank	invalid_bank_details	bacs	ARUDD-5ARUDD-YARUDD-E	The specified bank account does not exist or was closed.
bank	refer_to_payer	bacs	ARUDD-6	This mandate has been cancelled because a payment under it failed.
bank	bank_account_closed	bacs	AUDDIS-2	The bank account for this mandate has been closed as the customer is deceased.
bank	invalid_bank_details	bacs	AUDDIS-5	The specified bank account does not exist or was closed.
bank	bank_account_closed	bacs	AUDDIS-B	The customer's account was closed at their bank.
bank	direct_debit_not_enabled	bacs	AUDDIS-FAUDDIS-GAUDDIS-N	The bank account does not support Direct Debit.
bank	bank_account_transferred	bacs	AUDDIS-3AUDDIS-C	The customer's bank account was transferred to a different bank or building society.
bank	invalid_bank_details	becs	15	The bank account specified does not exist.
bank	direct_debit_not_enabled	becs	2	The bank account does not support Direct Debit.
bank	payment_stopped	becs	2	The payment was stopped by the payer or their bank.
bank	invalid_bank_details	becs	DEN	The bank account specified does not exist. Any subscriptions and pending payments will also be cancelled.
bank	other	becs	789	There was an internal error processing this mandate.
bank	invalid_bank_details	becs	RC02	The bank account specified does not exist.
bank	authorisation_disputed	becs	MD01	The customer disputes having authorised you to set up a mandate with them.
bank	bank_account_closed	becs	AC04	The customer's account was closed at their bank.
bank	bank_account_closed	becs	MD07	This bank account has been closed as the customer is deceased.
bank	invalid_bank_details	becs	AC01	The bank account specified does not exist. Any subscriptions and pending payments will also be cancelled.
bank	refer_to_payer	becs	NARR	There was an internal error processing this mandate.
bank	mandate_cancelled	pay_to	MD20	The mandate has expired.
bank	refer_to_payer	pay_to	NARRUNKN	There was an internal error processing this mandate.
bank	invalid_bank_details	becs_nz	DISHONOUR-AC05NEGATIVE_ACKNOWLEDGEMENT-AC05	The bank account specified does not exist. Any subscriptions and pending payments will also be cancelled.
bank	bank_account_closed	becs_nz	DISHONOUR-AC04	The customer's account was closed at their bank.
bank	invalid_bank_details	becs_nz	DISHONOUR-AC03	The customer's bank account was transferred to a different bank or building society.
bank	mandate_cancelled	becs_nz	DISHONOUR-AG01	This mandate was cancelled by the customer or their bank.
bank	other	becs_nz		The customer disputes having authorised you to set up a mandate with them.
bank	refer_to_payer	becs_nz		This mandate has been cancelled because a payment under it failed.
bank	invalid_bank_details	pad		The specified bank account does not exist or was closed.
bank	bank_account_closed	pad		The customer's account was closed at their bank.
bank	bank_account_closed	pad		This bank account has been closed as the customer is deceased.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

PAYOUTMENT\_STATUS\_REPORT\_V2

Origin	Cause	Scheme	Reason code	Description
bank	refer_to_payer	pad	PAYMENT_STATUS_REPORT-914	The account holder name may be different than that associated with the electronic transaction to run.
bank	invalid_bank_details	sepa_core	AC01	The bank account specified does not exist. Any subscriptions and pending payments will also be cancelled.

**transferred**

The mandate has been transferred to a different bank account either using a bank switching service (where it is supported) or with help from GoCardless Support when a customer asks to change their bank account (we can make the switch after verifying the details). The event will include `links[previous_customer_bank_account]` and `links[new_customer_bank_account]`. When using a bank switching service, the mandate may have been submitted again, depending on how the involved banks handled the transfer.

Origin	Cause	Description
--------	-------	-------------

bank	bank_account_transferred	The customer's bank account was transferred to a different bank or building society.
api	mandate_transferred	This mandate was transferred to a new bank account through GoCardless.

**expired**

No collection attempts were made against the mandate within the dormancy period of your service user number. As a result it has expired, and no further collections can be taken against it. If you wish to continue taking payments from this customer you should request their permission and use the [reinstate endpoint](#).

Origin	Cause	Description
gocardless	mandate_expired	The mandate is being marked as expired, because no payments have been collected against it for the dormancy period of your service user number. If you have access to the mandate reinstatement API endpoint, you can use this to attempt to set this mandate up again.
bank	mandate_cancelled	This mandate was cancelled by the customer or their bank.
gocardless	mandate_cancelled	The mandate has expired.

**submitted**

The mandate has been submitted to the banks, and should become active in a few days, unless the bank declines the request.

Origin	Cause	Description
gocardless	mandate_submitted	The mandate has been submitted to the banks.
bank	bank_account_transferred	The customer's bank account was transferred to a different bank or building society.

**resubmission\_requested**

A request to resubmit the mandate was made by the mandate [reinstate endpoint](#).

Origin	Cause	Description
api	resubmission_requested	An attempt to reinstate this mandate was requested.
bank	bank_account_transferred	The customer's bank account was transferred to a different bank or building society.

**reinstated**

The mandate has become active again, after it was cancelled or expired. This can be due to the customer's bank wishing to undo a cancellation or expiry notice they sent, or because the mandate was successfully reinstated via the [reinstate endpoint](#).

Origin	Cause	Description
gocardless	mandate_reinstated	The time window after submission for the banks to refuse a mandate has ended without any errors being received, so this mandate is now active.
bank	mandate_reinstated	A cancelled mandate has been re-instated by the customer's bank.

**replaced**

The mandate has been cancelled and replaced by a new mandate (for example, because the creditor has moved to a new Service User Number). The event will include `links[new_mandate]` with the ID of the new mandate.

Origin	Cause	Description
gocardless	scheme_identifier_change	The mandate has been cancelled and replaced by a new one.
consumed		
The mandate has been used to create a payment		
Origin	Cause	
gocardless	mandate_consumed	
blocked		
The mandate has been blocked because it cannot be unblocked and no payments can be created against it.		populated by you. This mandate cannot be unblocked and no payments can be created against it. If you still wish to collect payments from this customer, you will need to remove their details from

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

any blocks you have created and ask them to set up a new mandate. If you contacted GoCardless to block the customer's details, you will need to make a request to unlock them.

Origin	Cause	Description
gocardless	mandate_blocked	The mandate has been blocked because the customer's details matched against an entry in the blocklist populated by you.
gocardless	mandate_blocked_by_gocardless	The mandate has been blocked because the customer's details matched against an entry in our global blocklist.

HTTP  
HTTP

GET <https://api.gocardless.com/events/EV123> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV123",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "mandates",
    "action": "cancelled",
    "details": {
      "origin": "bank",
      "cause": "bank_account_closed",
      "description": "The bank account for this mandate has been closed.",
      "scheme": "bacs",
      "reason_code": "ADDACS-B"
    },
    "metadata": {},
    "resource_metadata": {
      "order_dispatch_date": "2014-05-22"
    },
    "links": {
      "mandate": "MD123"
    }
  }
}
```

GET <https://api.gocardless.com/events/EV456> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV456",
    "created_at": "2014-04-08T17:03:12.000Z",
    "resource_type": "mandates",
    "action": "cancelled",
    "details": {
      "origin": "api",
      "cause": "mandate_cancelled",
      "description": "The mandate was cancelled via an API call."
    },
    "metadata": {
      "cancelor_id": "some_id"
    },
    "resource_metadata": {
      "order_dispatch_date": "2014-05-22"
    },
    "links": {
      "mandate": "MD456"
    }
  }
}
```

GET <https://api.gocardless.com/events/EV789> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV789",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "mandates",
    "action": "transferred",
    "details": {
      "origin": "bank",
      "cause": "bank_account_transferred",
      "description": "The customer's bank account was transferred to a different bank or building society.",
      "scheme": "bacs",
      "reason_code": "AUDDIS-C"
    },
    "metadata": {},
    "resource_metadata": {
      "order_dispatch_date": "."
    },
    "links": {
      "mandate": "MD789",
      "previous_customer_bank_account": null,
      "new_customer_bank_account": null
    }
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Outbound Payment

This is a list of all the different values for action

<https://developer.gocardless.com/api-reference/>

275/322

**created**

The outbound payment has been created.

Origin	Cause	Description
api	outbound_payment_created	The outbound payment has been created.
gocardless	outbound_payment_created	The outbound payment has been created.

**cancelled**

The outbound payment has been cancelled.

Origin	Cause	Description
api	outbound_payment_cancelled	The outbound payment has been cancelled.
gocardless	outbound_payment_expired	The outbound payment has expired due to missing approval before the execution date.

**failed**

The outbound payment could not be sent, usually because the funding account did not have sufficient funds available.

Origin	Cause	Description
gocardless	outbound_payment_failed	The outbound payment has failed during processing.
gocardless	outbound_payment_rejected	The outbound payment has been rejected by the scheme or the bank.
bank	other	The payment failed but the reason for the failure was not provided.
gocardless	account_locked_for_payouts	Payments are not allowed from this account at the moment while we are running payouts from it.
bank	cancelled_for_regulatory_reasons	The payment was cancelled for regulatory reasons.
bank	forbidden_for_regulatory_reasons	The payment failed for regulatory reasons.
gocardless	insufficient_funds	The account had insufficient funds to make this payment.
gocardless	invalid_bank_details	The payment was cancelled because the recipient bank details are incorrect.
bank	invalid_bank_details	The payment was cancelled because the recipient bank details are incorrect.
gocardless	recipient_bank_not_reachable_for_scheme	Recipient bank account is not reachable for the chosen scheme.
bank	recipient_account_closed	The recipient bank account is closed.
bank	recipient_account_transferred	The recipient bank account has been transferred.
gocardless	recipient_CANNOT_ACCEPT_CURRENCY	Recipient bank account is unable to receive the specified currency.
bank	recipient_CANNOT_ACCEPT_CURRENCY	Recipient bank account is unable to receive the specified currency.
gocardless	recipient_is_sender	The recipient is the same as the sender.
gocardless	recoverable_error	The payment failed but the reason for the failure was not provided. Please, try again later.
gocardless	unrecoverable_error	The payment failed but the reason for the failure was not provided. Please contact support.
bank	sender_account_closed	The sender account is closed.
gocardless	internal_error	There was an internal error processing this payment.

**set\_to\_pending\_approval**

The outbound payment has been set for approval.

Origin	Cause	Description
gocardless	outbound_payment_set_to_pending_approval	The outbound payment has been set for approval.

**set\_to\_executing**

The outbound payment has begun execution.

Origin	Cause	Description
gocardless	outbound_payment_set_to_executing	The outbound payment has begun execution.

**executed**

The outbound payment has been confirmed by the scheme and the bank.

Origin	Cause	Description
gocardless	outbound_payment_executed	The outbound payment has been confirmed by the scheme and the bank.

HTTP  
HTTP

GET https://api.gocardless.com/

```
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "events": [
    {
      "id": "EV1234",
      "created_at": "2014-04-08T12:00:00Z",
      "resource_type": "outbound",
      "action": "created",
      "details": [
        {
          "origin": "api",
          "cause": "outbound_payment_created",
          "description": "The outbound payment has been created."
        }
      ],
      "links": [
        ...
      ]
    }
  ]
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "outbound_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    }
}
}
```

GET <https://api.gocardless.com/events/EV1237> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV1237",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "outbound_payments",
    "action": "cancelled",
    "details": {
      "origin": "api",
      "cause": "outbound_payment_cancelled",
      "description": "The outbound payment has been cancelled."
    },
    "metadata": {},
    "links": {
      "outbound_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    }
  }
}
```

GET <https://api.gocardless.com/events/EV1243> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV1237",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "outbound_payments",
    "action": "cancelled",
    "details": {
      "origin": "gocardless",
      "cause": "outbound_payment_expired",
      "description": "The outbound payment has expired due to missing approval before the execution date."
    },
    "metadata": {},
    "links": {
      "outbound_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    }
  }
}
```

GET <https://api.gocardless.com/events/EV1238> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV1238",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "outbound_payments",
    "action": "failed",
    "details": {
      "origin": "gocardless",
      "cause": "outbound_payment_failed",
      "description": "The outbound payment has failed during processing."
    },
    "metadata": {},
    "links": {
      "outbound_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    }
  }
}
```

GET <https://api.gocardless.com/events/EV1239> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV1239",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "outbound_payments",
    "action": "failed",
    "details": {
      "origin": "bank",
      "cause": "outbound_payment_rejected",
      "description": "The outbound payment has been rejected by the scheme or the bank."
    },
    "metadata": {},
    "links": {
      "outbound_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    }
  }
}
```

GET <https://api.gocardless.com/>

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": {
    "id": "EV1240",
    "created_at": "2014-04-08T17:01:06.000Z",
    "resource_type": "outbound_payments",
    "action": "cancelled"
  }
}
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "action": "set_to_pending_approval",
    "details": {
      "origin": "gocardless",
      "cause": "outbound_payment_set_to_pending_approval",
      "description": "The outbound payment has been set for approval.",
    },
    "metadata": {},
    "links": {
      "outbound_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"
    }
  }
}

```

GET https://api.gocardless.com/events/EV1241 HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json  
{  
 "events": {  
 "id": "EV1241",  
 "created\_at": "2014-04-08T17:01:06.000Z",  
 "resource\_type": "outbound\_payments",  
 "action": "set\_to\_executing",  
 "details": {  
 "origin": "gocardless",  
 "cause": "outbound\_payment\_set\_to\_executing",  
 "description": "The outbound payment has begun execution.",  
 },  
 "metadata": {},  
 "links": {  
 "outbound\_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"  
 }
 }
}

GET https://api.gocardless.com/events/EV1242 HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json  
{  
 "events": {  
 "id": "EV1242",  
 "created\_at": "2014-04-08T17:01:06.000Z",  
 "resource\_type": "outbound\_payments",  
 "action": "executed",  
 "details": {  
 "origin": "gocardless",  
 "cause": "outbound\_payment\_executed",  
 "description": "The outbound payment has been confirmed by the scheme and the bank."  
 },  
 "metadata": {},  
 "links": {  
 "outbound\_payment": "a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11"  
 }
 }
}

## Payer Authorisation

This is a list of all the different values for `action`

### completed

PayerAuthorisation is completed. Customer, CustomerBankAccount and Mandate have been created.

Origin	Cause	Description
gocardless	payer_authorisation_completed	PayerAuthorisation is completed. Customer, CustomerBankAccount and Mandate have been created.

### failed

PayerAuthorisation is failed. Customer, CustomerBankAccount and Mandate creation have been failed.

Origin	Cause	Description
gocardless	customer_creation_failed	PayerAuthorisation has failed. Customer, CustomerBankAccount and Mandate creation have failed.
gocardless	customer_bank_account_creation_failed	PayerAuthorisation has failed. Customer, CustomerBankAccount and Mandate creation have failed.
gocardless	mandate_creation_failed	PayerAuthorisation has failed. Customer, CustomerBankAccount and Mandate creation have failed.



HTTP

HTTP

GET https://api.gocardless.com/

## Payment

This is a list of all the different va

### created

The payment has been created.

Origin	Cause
api	payment_created

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

Consultez notre politique de confidentialité

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	payment_created	Payment created by a subscription.
api	instalment_schedule_created	Payment created by an instalment schedule.

**customer\_approval\_granted**

The payment required additional approval from the customer before it could be submitted, and that approval has been granted.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
customer	customer_approval_granted	The customer granted approval for this payment

**customer\_approval\_denied**

The payment required additional approval from the customer before it could be submitted, and that approval has been denied.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
customer	customer_approval_denied	The customer denied approval for this payment

**submitted**

The payment has been submitted to the banks. It will be a few days until it is collected, or fails.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	payment_submitted	Payment submitted to the banks. As a result, it can no longer be cancelled.

**confirmed**

The payment has been collected from the customer's bank account, and is now being held by GoCardless. It can take up to 5 working days for a payment to be collected, and will then be held for a short time before becoming paid\_out.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	payment_confirmed	Enough time has passed since the payment was submitted for the banks to return an error, so this payment is now confirmed.

**chargeback\_cancelled**

The customer's bank has cancelled the chargeback request. This is almost always at the request of the customer.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
bank	payment_confirmed	The chargeback for this payment was reversed
gocardless	payment_confirmed	The chargeback for this payment was reversed

**paid\_out**

The payment has left GoCardless and has been sent to the creditor's bank account.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	payment_paid_out	The payment has been paid out by GoCardless.

**late\_failure\_settled**

The payment was a late failure which had already been paid out, and has been debited from a payout.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	late_failure_settled	This late failed payment has been settled against a payout.

**chargeback\_settled**

The payment was charged back, having previously been paid out, and has been debited from a payout.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	chargeback_settled	This charged back payment has been settled against a payout.

**surcharge\_fee\_debited**

A surcharge fee has been charged for this payment, because it failed or got charged back.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
gocardless	surcharge_fee_debited	

**failed**

The payment could not be collected. This is usually caused by a missing or incorrect value in the will\_attempt\_retry field is true.

<b>Origin</b>	<b>Cause</b>	<b>Description</b>
bank	refer_to_payer	

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

callably retry the payment if event's

**Description**

The customer's account had insufficient funds to make this payment.

Origin	Cause	Scheme	Reason code	Description
bank	bank_account_closed	ach	ACH_RETURN-R14ACH_RETURN-R15	This payment failed because the customer is deceased.
bank	invalid_bank_details	ach	ACH_RETURN-R04ACH_RETURN-R13ACH_RETURN-R28ACH_RETURN-R82	The account number was invalid. The mandate will also be cancelled or failed.
bank	invalid_bank_details	ach	ACH_RETURN-R03	The bank account specified does not exist. The mandate will also be cancelled.
bank	bank_account_closed	ach	ACH_RETURN-R02	The bank account for this payment has been closed. The mandate will also be cancelled.
bank	bank_account_closed	ach	ACH_RETURN-R12	This payment has been cancelled because the account has been sold to another financial institution.
bank	refer_to_payer	ach	ACH_RETURN-R16	This payment has been cancelled because the payer's bank account is frozen. ACH authorization will be cancelled.
bank	refer_to_payer	ach	ACH_RETURN-R05ACH_RETURN-R32ACH_RETURN-R34ACH_RETURN-R83	The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.
bank	refer_to_payer	ach	ACH_RETURN-R20ACH_RETURN-R67ACH_RETURN-R75	The payment failed but the reason for the failure was not provided, usually for regulatory reasons.
bank	refer_to_payer	ach	ACH_RETURN-R08ACH_RETURN-R23ACH_RETURN-R29	The customer refused to accept this payment.
bank	authorisation_disputed	ach	ACH_RETURN-R07	The customer claims that they asked you to cancel their mandate before you took the payment.
bank	return_on_odfi_request	ach	ACH_RETURN-R06	This payment was charged back by the customer's bank, because the customer disputed authorising the transaction.
bank	authorisation_disputed	ach	ACH_RETURN-R06	This payment has been cancelled because the payer disputes authorising its mandate.
bank	authorisation_disputed	ach	ACH_RETURN-R11	The payment was charged back. Customer advises an authorization to debit exists, but there is an error or defect in the payment such that the entry does not conform to the terms of the authorization.
bank	authorisation_disputed	ach	ACH_RETURN-R05	This payment was charged back because the customer disputes having authorised you to set up a mandate with them.
bank	authorisation_disputed	ach	ACH_RETURN-R08	The customer has placed a stop on this payment and the authorisation has been cancelled. Please contact the customer to set up a new authorisation.
bank	authorisation_disputed	ach	ACH_RETURN-R31	This payment has been cancelled because the customer disputes authorising its mandate.
bank	authorisation_disputed	ach	ACH_RETURN-R10	The customer disputes having authorised you to set up a mandate with them.
bank	other	ach	ACH_RETURN-R24ACH_RETURN-R29ACH_RETURN-R83ACH_RETURN-R84ACH_RETURN-R85ACH_RETURN-R17	There was an internal error processing this payment.
bank	other	ach	ACH_RETURN-R81	This payment has been cancelled because the customer's bank does not support the required payment type or transaction format.
bank	other			This payment was cancelled because the return entry was not a duplicate of entry previously returned by the RDFI.
bank	other			A payment was cancelled because a financial institution's participation had been restricted by a regulatory authority.
bank	test_failure			GoCardless has marked this payment as failed in sandbox to enable testing of payment failure webhooks.
bank	refer_to_payer			The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	mandate_cancelled	bacs	ARUDD-1	The customer cancelled the mandate at their bank before the payment could be collected.
bank	bank_account_closed	bacs	ARUDD-2	This payment failed because the customer is deceased.
bank	bank_account_transferred	bacs	ARUDD-3	Your customer's mandate was transferred to a new bank account, but this payment was submitted to the old account. You may wish to retry the payment once you have received a transferred webhook for the corresponding mandate.
bank	authorisation_disputed	bacs	ARUDD-4	The customer has disputed having been notified of this Direct Debit.
bank	invalid_bank_details	bacs	ARUDD-5ARUDD-YARUDD-E	The account number was invalid. The mandate will also be cancelled or failed.
bank	bank_account_closed	bacs	ARUDD-B	The customer closed their account before the payment could be taken. The mandate will also be cancelled or failed.
bank	other	bacs	ARUDD-6	No mandate was setup for this payment. Some smaller banks require additional time to process mandates, and may not have processed this customer's mandate yet, so you may wish to retry the payment.
bank	authorisation_disputed	bacs	ARUDD-7	The customer has disputed that the amount taken differs from the amount they were notified of.
bank	invalid_bank_details	sepa_core	AC01BE06	The bank account specified does not exist. The mandate will also be cancelled.
bank	bank_account_closed	sepa_core	AC04	The bank account for this payment has been closed. The mandate will also be cancelled.
bank	direct_debit_not_enabled	sepa_core	AG01AC06	The bank account for this payment does not support SEPA Direct Debit. The mandate will also be cancelled.
bank	account_blocked_for_any_financial_transaction	sepa_core	AC06	This payment failed because the payer's account was blocked.
bank	insufficient_funds	sepa_core	AM04ED05	The customer's account had insufficient funds to make this payment.
bank	mandate_cancelled	sepa_core	MD01	The customer cancelled their mandate at their bank.
bank	bank_account_closed	sepa_core	MD07	This payment failed because the customer is deceased.
bank	refer_to_payer	sepa_core	MS03RR04	The payment failed but the reason for the failure was not provided, usually for regulatory reasons.
bank	refer_to_payer	sepa_core	MS02	The customer refused to accept this payment.
bank	refer_to_payer	sepa_core	SL01	The payment failed due to a restriction on Direct Debit payments from the payer's bank account.
bank	other	becs	9	The customer's bank refused to accept this payment, please refer to customer.
bank	other	becs	278	There was an internal error processing this payment.
bank	payment_stopped	becs	2	The payment was stopped by the payer or their bank.
bank	direct_debit_not_enabled		-----	The payment failed due to a restriction on Direct Debit payments from the payer's bank account.
bank	refer_to_payer		-----	The customer's bank wasn't able to do the Direct Debit. This is almost always due to insufficient funds, but occasionally used as a catch-all for other failures.
bank	invalid_bank_details		-----	account number was invalid. The mandate will also be cancelled or failed.
bank	bank_account_closed		-----	The bank account for this payment has been closed. The mandate will also be cancelled.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	invalid_bank_details	becs	DEN	The account number was invalid. The mandate will also be cancelled or failed.
gocardless	other	becs	INT_ERR	There was an internal error processing this payment.
bank	invalid_bank_details	becs	RC02	The bank account specified does not exist. The mandate will also be cancelled.
bank	authorisation_disputed	becs	MD01	The customer disputes having authorised you to set up a mandate with them.
bank	bank_account_closed	becs	AC04	The bank account for this payment has been closed. The mandate will also be cancelled.
bank	bank_account_closed	becs	MD07	This payment failed because the customer is deceased.
bank	invalid_bank_details	becs	AC01	The account number was invalid. The mandate will also be cancelled or failed.
bank	refer_to_payer	becs	NARR	The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.
bank	other	pay_to	278	There was an internal error processing this payment.
bank	invalid_bank_details	pay_to	15	The account number was invalid. The mandate will also be cancelled or failed.
bank	bank_account_closed	pay_to	34	The bank account for this payment has been closed. The mandate will also be cancelled.
bank	invalid_bank_details	pay_to	RC02	The bank account specified does not exist. The mandate will also be cancelled.
bank	refer_to_payer	betalingservice	PAYMENT_INFORMATION-0237	The customer refused to accept this payment.
bank	mandate_cancelled	betalingservice	PAYMENT_INFORMATION-0238	This payment was canceled because the customer cancelled the mandate at their bank.
bank	refer_to_payer	betalingservice	INFORMATION_LIST-0282	The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.
bank	other	betalingservice	INFORMATION_LIST-0253	This payment was canceled because the customer cancelled the mandate at their bank.
bank	invalid_bank_details	becs_nz	DISHONOUR-AC05NEGATIVE_ACKNOWLEDGEMENT-AC05	The bank account specified does not exist. The mandate will also be cancelled.
bank	bank_account_closed	becs_nz	DISHONOUR-AC04	The bank account for this payment has been closed. The mandate will also be cancelled.
bank	mandate_cancelled	becs_nz	DISHONOUR-AG01	This payment was canceled because the customer cancelled the mandate at their bank.
bank	invalid_bank_details	becs_nz	DISHONOUR-AC03NEGATIVE_ACKNOWLEDGEMENT-AC03	The bank account specified does not exist. The mandate will also be cancelled.
bank	refer_to_payer	becs_nz	DISHONOUR-AM04NEGATIVE_ACKNOWLEDGEMENT-AM04	The customer's account had insufficient funds to make this payment.
bank	refer_to_payer	becs_nz	DISHONOUR-MS01	The customer refused to accept this payment.
bank	refer_to_payer			The payment failed due to a restriction on Direct Debit payments from the payer's bank account.
bank	refer_to_payer			The customer's account had insufficient funds to make this payment.
bank	invalid_bank_details			account number was invalid. The mandate will also be cancelled or failed.
bank	bank_account_closed			The bank account for this payment has been closed. The mandate will

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	refer_to_payer	pad	PAYMENT_STATUS_REPORT-907PAYMENT_STATUS_REPORT-911PAYMENT_STATUS_REPORT-990	The payment failed due to a restriction on Direct Debit payments from the payer's bank account.
bank	bank_account_closed	pad	PAYMENT_STATUS_REPORT-910	This payment failed because the customer is deceased.
bank	refer_to_payer	pad	PAYMENT_STATUS_REPORT-903PAYMENT_STATUS_REPORT-908PAYMENT_STATUS_REPORT-914	The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.
bank	other	pad	PAYMENT_STATUS_REPORT-909	The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.
bank	other	becs_nz	DISHONOUR-BE05	The customer disputes having authorised you to set up a mandate with them.
bank	other	becs_nz	ACKNOWLEDGEMENT-FAILED	The payment failed but the reason for the failure was not provided, usually for regulatory reasons.
bank	other	faster_payments	FAILED	The payment failed but the reason for the failure was not provided, usually for regulatory reasons.
bank	insufficient_funds	faster_payments	FAILED	The customer's account had insufficient funds to make this payment.
bank	bank_declined_payment	faster_payments	FAILED	Payment declined by the bank.
bank	daily_payment_limit_reached	faster_payments	FAILED	Payment exceeds the daily payment limit for this payer, imposed by the bank.
bank	insufficient_payment_permissions	faster_payments	FAILED	Payment denied due to missing approvals from the bank.
bank	payment_violated_mandate_parameters	faster_payments	FAILED	The payment failed due to a violation of the associated mandate consent parameters.
bank	other	sepa_credit_transfer	FAILED	The payment failed but the reason for the failure was not provided, usually for regulatory reasons.
bank	insufficient_funds	sepa_credit_transfer	FAILED	The customer's account had insufficient funds to make this payment.
bank	other	sepa_instant_credit_transfer	FAILED	The payment failed but the reason for the failure was not provided, usually for regulatory reasons.
bank	insufficient_funds	sepa_instant_credit_transfer	FAILED	The customer's account had insufficient funds to make this payment.
bank	other	pay_to	FAILED	The payment failed but the reason for the failure was not provided, usually for regulatory reasons.
bank	insufficient_funds	pay_to	FAILED	The customer's account had insufficient funds to make this payment.
bank	closed_debtor_account_number	pay_to	FAILED	This payment failed because the payer's account was closed.
bank	account_details_changed	pay_to	FAILED	This payment failed because the payer's account was changed to a different account.
bank	blocked_account	pay_to	FAILED	This payment failed because the payer's account was blocked.
bank	unsuccessful_direct_debit			This payment could not be completed due to an unsuccessful transaction against the mandate.
bank	limit_exceeded			A payment exceeds the maximum limit set by the bank for this payer.
bank	mandate_expired			The mandate for this payment has expired.
bank	bank_system_processing_error			The customer's bank refused to accept this payment, please refer to the customer.
bank	maximum_number_of_dd_transactions_exceeded			The payment failed due to a violation of the associated mandate consent parameters.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	maximum_dd_transaction_amount_exceeded	pay_to	FAILED	The payment failed due to a violation of the associated mandate consent parameters.
bank	payment_violated_mandate_parameters	pay_to	FAILED	The payment failed due to a violation of the associated mandate consent parameters.
gocardless	no_status_update_faster_payments		FAILED	Payment automatically failed after 24h without status update from bank

**charged\_back**

The customer asked their bank to refund the payment under the Direct Debit Guarantee, and it has been returned to the customer.

Origin	Cause	Scheme	Reason code	Description
bank	authorisation_disputed	ach	ACH_RETURN-R05ACH_RETURN-R31	This payment was charged back by the customer's bank, because the customer disputed authorising the transaction.
bank	refund_requested	ach	ACH_RETURN-R16	This payment was charged back by the customer's bank at the customer's request within the 8 week cool-off period.
bank	mandate_cancelled	ach	ACH_RETURN-R07	This payment was charged back because the mandate was withdrawn.
bank	authorisation_disputed	ach	ACH_RETURN-R08ACH_RETURN-R10ACH_RETURN-R11ACH_RETURN-R29	The customer disputes having authorised you to set up a mandate with them.
bank	return_on_odfi_request	ach	ACH_RETURN-R06	This payment was charged back by the customer's bank, because the customer disputed authorising the transaction.
bank	authorisation_disputed	bacs	DDICA-1	The customer has disputed that the amount taken differs from the amount they were notified of.
bank	authorisation_disputed	bacs	DDICA-2	The customer has disputed having been notified of this Direct Debit.
bank	authorisation_disputed	bacs	DDICA-4	The customer claims that they asked you to cancel their mandate before you took the payment.
bank	authorisation_disputed	bacs	DDICA-5DDICA-6DDICA-8	The customer disputes having authorised you to set up a mandate with them.
bank	mandate_cancelled	sepa_core	MD01	This payment was charged back by the customer's bank, because the customer disputed authorising the transaction.
bank	refund_requested	sepa_core	MD06	This payment was charged back by the customer's bank at the customer's request within the 8 week cool-off period.
bank	refund_requested	betalingsservice	PAYMENT_INFORMATION-0239	This payment was charged back by the customer's bank at the customer's request within the 8 week cool-off period.
bank	authorisation_disputed	becs	UDUNAUT	The customer disputes having authorised you to set up a mandate with them.
bank	mandate_cancelled	becs	CBC	The customer cancelled their mandate at their bank.
bank	other	becs	UCDUOTHR	The payment was charged back.
bank	authorisation_disputed	becs	MD01	The customer disputes having authorised you to set up a mandate with them.
bank	authorisation_disputed	pad	PAYMENT_STATUS_REPORT-915	The customer disputes having authorised you to set up a mandate with them.
bank	other	pad	PAYMENT_STATUS_REPORT-916PAYMENT_STATUS_REPORT-919	The customer has disputed that the amount taken differs from the amount they were notified of.
bank	other	pad	PAYMENT_STATUS_REPORT-918PAYMENT_STATUS_REPORT-921	The customer has disputed having been notified of this Direct Debit.
bank	mandate_cancelled	pad	PAYMENT_STATUS_REPORT-917PAYMENT_STATUS_REPORT-920	This payment was charged back because the mandate was withdrawn.

cancelled

The payment was cancelled.

Origin	Cause	Scheme	Reason code	Description
bank	refer_to_payer	ach	ACH_RETURN-R20ACH_RETURN-R34	This payment has been cancelled because the bank account it was going to be taken from does not support Direct Debit.
bank	authorisation_disputed	ach	ACH_RETURN-R05ACH_RETURN-R06ACH_RETURN-R08ACH_RETURN-R10ACH_RETURN-R11ACH_RETURN-R29ACH_RETURN-R31ACH_RETURN-R51	This payment has been cancelled because the payer disputes authorising its mandate.
bank	refer_to_payer	ach	ACH_RETURN-R29	This payment has been cancelled because its mandate was cancelled.
bank	mandate_cancelled	ach	ACH_RETURN-R07ACH_RETURN-R06	This payment has been cancelled because its mandate was cancelled.
bank	other	ach	ACH_RETURN-R85	There was an internal error processing this payment.
gocardless	instalment_schedule_cancelled	-	-	payment_cancelled_at_request
api	payment_cancelled	-	-	This payment was cancelled at your request.
api	mandate_cancelled	-	-	The mandate for this payment was cancelled at your request.
api	instalment_schedule_cancelled	-	-	payment_cancelled_at_request
api	bank_account_closed	-	-	The bank account for this payment was disabled at your request.
bank	mandate_cancelled	bacs	ADDACS-0ADDACS-1	The mandate for this payment was cancelled at a bank branch.
bank	bank_account_closed	bacs	ADDACS-2ADDACS-B	The mandate for this payment was cancelled as the customer's bank account has been closed.
bank	bank_account_transferred	bacs	ADDACS-3ADDACS-C	The mandate for this payment was cancelled as the customer asked their bank to transfer the mandate to a new account, but the bank has failed to send GoCardless the new bank details.
bank	authorisation_disputed	bacs	ADDACS-D	The customer disputes authorising this mandate.
bank	bank_account_closed	bacs	ARUDD-2	This payment was cancelled because the customer is deceased.
bank	bank_account_closed	bacs	ARUDD-B	This payment was cancelled because the customer closed their bank account before it could be collected.
bank	refer_to_payer	bacs	ARUDD-6	This payment was canceled because the customer cancelled the mandate at their bank.
bank	invalid_bank_details	sepa_core	AC01BE06	This payment has been cancelled because the bank details for its mandate are incorrect.
bank	bank_account_closed	sepa_core	AC04MD07	This payment has been cancelled because the bank account it was going to be taken from has been closed.
bank	direct_debit_not_enabled	sepa_core	AG01AC06	This payment has been cancelled because the bank account it was going to be taken from does not support Direct Debit.
bank	account_blocked_for_any_financial_transaction	sepa_core	AC06	This payment failed because the payer's account was blocked.
bank	authorisation_disputed			This payment has been cancelled because the payer disputes authorising its mandate.
bank	mandate_cancelled			This payment has been cancelled because the payer disputes authorising its mandate.
bank	authorisation_disputed			This payment has been cancelled because the customer disputes authorising its mandate.
bank	bank_account_closed			This payment has been cancelled because the bank account it was going to be taken from has been closed.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	invalid_bank_details	bacs	AUDDIS-5	This payment has been cancelled because the bank details for its mandate are not valid.
bank	bank_account_transferred	bacs	AUDDIS-3AUDDIS-C	This payment has been cancelled because the customer requested their bank transfer their Direct Debit to a new account but the bank did not send GoCardless the new bank details.
bank	direct_debit_not_enabled	bacs	AUDDIS-FAUDDIS-GAUDDIS-N	This payment has been cancelled because the bank account for its mandate does not support Direct Debit.
bank	invalid_bank_details	bacs	ARUDD-5ARUDD-EARUDD-Y	This payment has been cancelled because the bank details for its mandate are not valid.
bank	mandate_cancelled	bacs	ARUDD-1	This payment was canceled because the customer cancelled the mandate at their bank.
bank	mandate_cancelled	betalingsservice	PAYMENT_INFORMATION-0238	This payment was canceled because the customer cancelled the mandate at their bank.
bank	mandate_cancelled	betalingsservice	MANDATE_INFORMATION-0232MANDATE_INFORMATION-0233MANDATE_INFORMATION-0234	This payment was canceled because the customer cancelled the mandate at their bank.
bank	mandate_cancelled	betalingsservice	INFORMATION_LIST-0253INFORMATION_LIST-0257	This payment was canceled because the customer cancelled the mandate at their bank.
bank	invalid_bank_details	becs_nz	DISHONOUR-AC03	This payment has been cancelled because the customer requested their bank transfer their Direct Debit to a new account but the bank did not send GoCardless the new bank details.
bank	refer_to_payer	becs_nz	DISHONOUR-MS01	This payment was canceled because the customer cancelled the mandate at their bank.
bank	bank_account_closed	becs_nz	DISHONOUR-AC04	The bank account for this payment has been closed. The mandate will also be cancelled.
bank	invalid_bank_details	becs_nz	DISHONOUR-AC05NEGATIVE_ACKNOWLEDGEMENT-AC05	The bank account specified does not exist. The mandate will also be cancelled.
bank	mandate_cancelled	becs_nz	DISHONOUR-AG01	This payment was canceled because the customer cancelled the mandate at their bank.
bank	other	becs_nz	DISHONOUR-BE05	The customer disputes having authorised you to set up a mandate with them.
bank	invalid_bank_details	pad	PAYMENT_STATUS_REPORT-900PAYMENT_STATUS_REPORT-902PAYMENT_STATUS_REPORT-912PAYMENT_STATUS_REPORT-1023PAYMENT_STATUS_REPORT-2014PAYMENT_STATUS_REPORT-2017PAYMENT_STATUS_REPORT-2018PAYMENT_STATUS_REPORT-2019PAYMENT_STATUS_REPORT-2020PAYMENT_STATUS_REPORT-2034PAYMENT_STATUS_REPORT-0518PAYMENT_STATUS_REPORT-0567PAYMENT_STATUS_REPORT-0573	This payment has been cancelled because the bank details for its mandate are incorrect.
bank	bank_account_closed	pad	PAYMENT_STATUS_REPORT-905	This payment has been cancelled because the bank account it was going to be taken from has been closed.
bank	bank_account_closed	pad	PAYMENT_STATUS_REPORT-910	This payment was cancelled because the customer is deceased.
bank	other	pad	PAYMENT_STATUS_REPORT-918PAYMENT_STATUS_REPORT-921	This payment has been cancelled because the customer disputes authorising its mandate.
bank	authorisation_disputed			This payment has been cancelled because the customer disputes authorising its mandate.
bank	mandate_cancelled			This mandate was cancelled by the customer or their bank.
bank	mandate_cancelled			This mandate was cancelled by the customer or their bank.
bank	invalid_bank_details			This payment has been cancelled because the bank details for its mandate are incorrect.
bank	direct_debit_not_enabled			This payment has been cancelled because the bank account for its mandate does not support Direct Debit.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	payment_stopped	becs	2	The payment was stopped by the payer or their bank.
bank	bank_account_closed	becs	3	This payment was canceled because the customer cancelled the mandate at their bank.
bank	bank_account_closed	becs	4	This payment was canceled because the customer cancelled the mandate at their bank.
bank	invalid_bank_details	becs	DEN	This payment has been cancelled because the bank details for its mandate are incorrect.
bank	other	becs	789	An error was received from the banks while setting up the mandate for this payment.
bank	authorisation_disputed	becs	UDUNAUT	The customer disputes having authorised you to set up a mandate with them.
bank	mandate_cancelled	becs	CBC	This payment has been cancelled because its mandate was cancelled.
bank	invalid_bank_details	becs	RC02	This payment has been cancelled because the bank details for its mandate are incorrect.
bank	authorisation_disputed	becs	MD01	This payment has been cancelled because the customer disputes authorising its mandate.
bank	bank_account_closed	becs	AC04	This payment has been cancelled because the bank account it was going to be taken from has been closed.
bank	bank_account_closed	becs	MD07	This payment was cancelled because the customer is deceased.
bank	invalid_bank_details	becs	AC01	This payment has been cancelled because the bank details for its mandate are incorrect.
bank	refer_to_payer	becs	NARR	The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.
bank	invalid_bank_details	pay_to	15	This payment has been cancelled because the bank details for its mandate are incorrect.
bank	bank_account_closed	pay_to	3	This payment was canceled because the customer cancelled the mandate at their bank.
bank	invalid_bank_details	pay_to	DEN	This payment has been cancelled because the bank details for its mandate are incorrect.
gocardless	mandate_expired	-	-	The mandate expired before the payment could be collected.
bank	refer_to_payer	pad	PAYMENT_STATUS_REPORT-903PAYMENT_STATUS_REPORT-908PAYMENT_STATUS_REPORT-914	The customer's bank wasn't able to pay the Direct Debit. This is almost always due to insufficient funds, but is occasionally used as a catch-all for other failures.
bank	mandate_cancelled	faster_payments	PAYER_MANDATE_CANCELLED	This payment was canceled because the customer cancelled the mandate at their bank.
bank	mandate_cancelled	faster_payments	MANDATE_CANCELLED_BY_BANK	This payment was canceled because the customer cancelled the mandate at their bank.
bank	mandate_cancelled	faster_payments	FAILED_ELIGIBILITY_CHECK	This payment was canceled because the customer cancelled the mandate at their bank.
bank	mandate_suspended_by_payer			The mandate for this payment was suspended by the payer.
gocardless	mandate_cancelled			The mandate for this payment was cancelled at your request.
gocardless	mandate_cancelled			The mandate expired before the payment could be collected.
bank	return_on_odfi_request			The payment was cancelled because of an ODFI return request.
bank	refer_to_payer			This payment has been cancelled because the payer's bank account is frozen. ACH authorization will be cancelled.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
gocardless	initial_one_off_payment_failed	-	-	This mandate has been cancelled because the initial faster payment failed

**resubmission\_requested**

A request to resubmit the payment was made by the payment [retry endpoint](#). This can also mean that the payment was automatically scheduled for resubmission by GoCardless, if you have opted in for the [intelligent retries](#) feature.

Origin	Cause	Description
api	payment_retried	An attempt to retry this payment was requested.
gocardless	payment_autoretried	The payment was scheduled for resubmission automatically by GoCardless.

**sds\_settlement\_delayed**

We are withholding this payment from same-day settlement due to a higher possibility of failure.

Origin	Cause	Description
gocardless	sds_settlement_delayed	We are withholding this payment from same-day settlement due to a higher possibility of failure.



HTTP  
HTTP

```
GET https://api.gocardless.com/events/EV1234 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payments",
      "action": "failed",
      "details": {
        "origin": "bank",
        "cause": "insufficient_funds",
        "description": "The customer's account had insufficient funds to make this payment.",
        "scheme": "sepa_core",
        "reason_code": "AM04",
        "will_attempt_retry": false
      },
      "metadata": {},
      "resource_metadata": {
        "order_dispatch_date": "2014-05-22"
      },
      "links": {
        "payment": "PM123"
      }
    }
  ]
}
```

```
GET https://api.gocardless.com/events/EV4567 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "events": [
    {
      "id": "EV4567",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payments",
      "action": "cancelled",
      "details": {
        "origin": "bank",
        "cause": "mandate_cancelled",
        "description": "The mandate for this payment was cancelled as the customer's bank account has been closed.",
        "scheme": "bacs",
        "reason_code": "ADDACS-B"
      },
      "metadata": {},
      "resource_metadata": {
        "order_dispatch_date": "2014-05-22"
      },
      "links": {
        "payment": "PM456",
        "parent_event": "EV789"
      }
    }
  ]
}
```

```
GET https://api.gocardless.com/
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "events": [
    {
      "id": "EV8901",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payments",
      "action": "cancelled",
      "details": {
        "origin": "api",
        "cause": "payment_cancelled",
        "description": "Payment cancelled by customer"
      }
    }
  ]
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "metadata": {
      "cancelor_id": "some_id"
    },
    "resource_metadata": {
      "order_dispatch_date": "2014-05-22"
    },
    "links": {
      "payment": "PM789"
    }
  }
}

```

GET <https://api.gocardless.com/events/EV2345> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": [
    {
      "id": "EV2345",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payments",
      "action": "confirmed",
      "details": {
        "origin": "gocardless",
        "cause": "payment_confirmed",
        "description": "Payment confirmed"
      },
      "metadata": {},
      "resource_metadata": {
        "order_dispatch_date": "2014-05-22"
      },
      "links": {
        "payment": "PM999"
      }
    }
  ]
}
```

## Payout

This is a list of all the different values for `action`

### **paid**

GoCardless has transferred the payout to the creditor's bank account. The `details[cause]` will always be `payout_paid`, and the `details[origin]` will be `gocardless`.

Origin	Cause	Description
gocardless	payout_paid	GoCardless has transferred the payout to the creditor's bank account. FX payouts will emit this event but will continue to have a pending status until we emit the <code>payout_fx_rate_confirmed</code> event.

### **fx\_rate\_confirmed**

The exchange rate for the payout has been confirmed and will not change.

Origin	Cause	Description
gocardless	payout_fx_rate_confirmed	The exchange rate for the payout has been confirmed and will not change. Only sent for FX payouts.

### **tax\_exchange\_rates\_confirmed**

The tax exchange rates for all payout items of the payout have been confirmed. This event will only exist if the payout has a `tax_currency` and if its `tax_currency` is different from its `currency`. It will be created once all fees in the payout are invoiced.

Origin	Cause	Description
gocardless	payout_tax_exchange_rates_confirmed	The exchange rates for the taxes (such as VAT) that applied to GoCardless fees deducted from the payout have all been confirmed and will not change.



HTTP  
HTTP

GET <https://api.gocardless.com/events/EV123> HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json

```
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payouts",
      "action": "paid",
      "details": {
        "origin": "gocardless",
        "cause": "payout_paid",
        "description": "Payout sent"
      },
      "metadata": {},
      "resource_metadata": {
        "order_dispatch_date": "2014-05-22"
      },
      "links": {
        "payout": "PO123"
      }
    }
  ]
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Refund

This is a list of all the different values for `action`

### created

A refund has been created. The `details[cause]` will be `payment_refunded`, and the `details[origin]` will be `api`.

Origin	Cause	Description
--------	-------	-------------

<code>api</code>	<code>payment_refunded</code>	The refund has been created, and will be submitted in the next batch.
<code>api</code>	<code>refund_created</code>	The refund has been created, and will be submitted in the next batch.

### failed

The refund did not reach your customer, the funds will be returned to you. The `details[cause]` will be `refund_failed`, and the `details[origin]` will be `gocardless`. It is important to note that the ‘failed’ and ‘refund\_settled’ events are not associated. A refund can fail even after it’s been settled. If a refund fails, GoCardless has attempted to refund the customer and has already deducted the applicable funds from one of your payouts. The refund has then subsequently failed to reach the customer. If this occurs, the funds will be returned to you.

Origin	Cause	Description
--------	-------	-------------

<code>gocardless</code>	<code>refund_failed</code>	The refund did not reach your customer, the funds will be returned to you.
-------------------------	----------------------------	--

### paid

The refund has been paid to your customer. The `details[cause]` will be `refund_paid`, and the `details[origin]` will be `gocardless`.

Origin	Cause	Description
--------	-------	-------------

<code>gocardless</code>	<code>refund_paid</code>	The refund has been paid to your customer.
-------------------------	--------------------------	--

### refund\_settled

The refund has been deducted from a payout. The `details[cause]` will be `refund_settled`, and the `details[origin]` will be `gocardless`.

Origin	Cause	Description
--------	-------	-------------

<code>gocardless</code>	<code>refund_settled</code>	The refund has been deducted from a payout.
-------------------------	-----------------------------	---

### funds\_returned

The refund has been credited in a payout. The `details[cause]` will be `refund_funds_returned`, and the `details[origin]` will be `gocardless`.

Origin	Cause	Description
--------	-------	-------------

<code>gocardless</code>	<code>refund_funds_returned</code>	The funds for the refund have been returned to you.
-------------------------	------------------------------------	---



HTTP  
HTTP

```
GET https://api.gocardless.com/events/EV123 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "refunds",
      "action": "created",
      "details": {
        "origin": "api",
        "cause": "payment_refunded",
        "description": "The refund has been created, and will be submitted in the next batch."
      },
      "metadata": {},
      "resource_metadata": {
        "order_dispatch_date": "2014-05-22"
      },
      "links": {
        "refund": "RF123"
      }
    }
  ]
}
```

## Scheme Identifier

This is a list of all the different va

### activated

This scheme identifier has been ac

Origin	Cause
--------	-------

<code>gocardless</code>	<code>scheme_identifier_act</code>
-------------------------	------------------------------------



HTTP  
HTTP

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
GET https://api.gocardless.com/events/EV123 HTTP/1.1
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2023-04-08T17:01:06.000Z",
      "resource_type": "scheme_identifiers",
      "action": "activated",
      "links": [
        {
          "scheme_identifier": "SU222"
        }
      ],
      "details": [
        {
          "origin": "gocardless",
          "cause": "scheme_identifier_activated",
          "description": "This scheme identifier has been activated."
        }
      ],
      "metadata": {}
    }
  ]
}
```

## Subscription

This is a list of all the different values for `action`

### **created**

The subscription has been created.

Origin	Cause	Description
api	subscription_created	Subscription created via the API.

### **customer\_approval\_granted**

The subscription required additional approval from the customer before it could become active, and that approval has been granted.

Origin	Cause	Description
customer	customer_approval_granted	The customer granted approval for this subscription

### **customer\_approval\_denied**

The subscription required additional approval from the customer before it could become active, and that approval has been denied.

Origin	Cause	Description
customer	customer_approval_denied	The customer denied approval for this subscription

### **amended**

The subscription amount has been changed.

Origin	Cause	Description
api	subscription_amended	Subscription amount has been amended.

### **payment\_created**

A payment has been created by this subscription.

Origin	Cause	Description
gocardless	payment_created	Payment created by a subscription.

### **cancelled**

This subscription has been cancelled. No further payments will be created.

Origin	Cause	Scheme	Reason code	Description
bank	bank_account_closed	ach	ACH_RETURN-R14ACH_RETURN-R15	This subscription was cancelled because the customer is deceased.
bank	return_on_offi_request	ach	ACH_RETURN-R06	This subscription has been cancelled because its mandate was cancelled.
bank	refer_to_payer			This subscription has been cancelled because the bank details for its mandate are incorrect.
bank	bank_account_closed			This subscription has been cancelled because the bank account was going to be taken from has been closed.
bank	mandate_cancelled			This subscription was canceled cause the customer cancelled the mandate at their bank.

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

 Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	mandate_cancelled	ach	ACH_RETURN-R10	This subscription has been cancelled because the bank details for its mandate are incorrect.
bank	invalid_bank_details	ach	ACH_RETURN-R03ACH_RETURN-R04ACH_RETURN-R13ACH_RETURN-R28ACH_RETURN-R82	This subscription has been cancelled because the bank details for its mandate are incorrect.
bank	direct_debit_not_enabled	ach	ACH_RETURN-R16ACH_RETURN-R20ACH_RETURN-R34	This subscription has been cancelled because the bank account it was going to be taken from does not support direct debit.
bank	refer_to_payer	ach	ACH_RETURN-R16ACH_RETURN-R20ACH_RETURN-R29	This subscription has been cancelled because its mandate was cancelled.
bank	authorisation_disputed	ach	ACH_RETURN-R05ACH_RETURN-R08ACH_RETURN-R10ACH_RETURN-R11ACH_RETURN-R29ACH_RETURN-R31ACH_RETURN-R51	This subscription has been cancelled because the customer disputes authorising its mandate.
api	subscription_cancelled	-	-	The subscription has been cancelled at your request.
api	mandate_cancelled	-	-	The subscription was cancelled because its mandate was cancelled at your request.
api	bank_account_closed	-	-	The mandate for this subscription was cancelled at your request.
gocardless	mandate_expired	-	-	The subscription was cancelled because its mandate has expired.
bank	mandate_cancelled	bacs	ADDACS-0ADDACS-1	The mandate for this subscription was cancelled at a bank branch.
bank	bank_account_closed	bacs	ADDACS-2ADDACS-B	The mandate for this subscription was cancelled as the customer's bank account has been closed.
bank	bank_account_transferred	bacs	ADDACS-3ADDACS-CADDACS-E	The mandate for this subscription was cancelled as the customer asked their bank to transfer the mandate to a new account, but the bank has failed to send GoCardless the new bank details.
bank	authorisation_disputed	bacs	ADDACS-D	The customer disputes authorising this mandate.
bank	bank_account_closed	bacs	ARUDD-2	This subscription was cancelled because the customer is deceased.
bank	refer_to_payer	bacs	ARUDD-6	An error was received from the banks while setting up the mandate for this subscription.
bank	bank_account_closed	bacs	ARUDD-B	This subscription was cancelled because the customer closed their bank account before it could be collected.
bank	invalid_bank_details	sepa_core	AC01BE06	This subscription has been cancelled because the bank details for its mandate are incorrect.
bank	bank_account_closed	sepa_core	AC04MD07	This subscription has been cancelled because the bank account it was going to be taken from has been closed.
bank	direct_debit_not_enabled	sepa_core	AG01AC06	This subscription has been cancelled because the bank account it was going to be taken from does not support direct debit.
bank	account_blocked_for_any_financial_transaction	sepa_core	AC06	This subscription has been cancelled because the bank account for its mandate was blocked.
bank	authorisation_disputed	sepa_core	MD01	This subscription has been cancelled because the payer disputes authorising its mandate.
bank	mandate_cancelled	sep	-	This subscription has been cancelled because the payer disputes authorising its mandate.
bank	authorisation_disputed	-	-	This subscription has been cancelled because the customer disputes authorising its mandate.
bank	bank_account_closed	-	-	This subscription has been cancelled because the bank account or its mandate has been closed.
bank	invalid_bank_details	-	-	This subscription has been cancelled because the bank details for its mandate are not valid.
bank	bank_account_transferred	-	-	This subscription has been cancelled because the customer has requested for direct debits to be

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Origin	Cause	Scheme	Reason code	Description
bank	direct_debit_not_enabled	bacs	AUDDIS-FAUDDIS-GAUDDIS-N	taken from a new account, but their bank did not send GoCardless the new bank details.
bank	invalid_bank_details	bacs	ARUDD-5ARUDD-YARUDD-E	This subscription has been cancelled because the bank account for its mandate does not support direct debit.
bank	mandate_cancelled	bacs	ARUDD-1	This subscription was canceled because the customer cancelled the mandate at their bank.
api	plan_cancelled	-	-	The subscription has been cancelled because the associated plan was cancelled.
bank	authorisation_disputed	beecs	UDUNAUT	This subscription has been cancelled because the customer disputes authorising its mandate.
bank	mandate_cancelled	beecs	CBC	This subscription has been cancelled because its mandate was cancelled.
bank	direct_debit_not_enabled	beecs	2	This subscription has been cancelled because the bank account for its mandate does not support direct debit.
bank	payment_stopped	beecs	2	The subscription was cancelled because the payment was stopped by the payer or their bank.
bank	bank_account_closed	beecs	4	This subscription was cancelled because the customer is deceased.
bank	invalid_bank_details	beecs	DEN	This subscription has been cancelled because the bank details for its mandate are not valid.
bank	bank_account_closed	beecs_nz	DISHONOUR-AC04	This subscription has been cancelled because the bank account it was going to be taken from has been closed.
bank	invalid_bank_details	pad	PAYMENT_STATUS_REPORT-900PAYMENT_STATUS_REPORT-902PAYMENT_STATUS_REPORT-912PAYMENT_STATUS_REPORT-1023PAYMENT_STATUS_REPORT-2014PAYMENT_STATUS_REPORT-2017PAYMENT_STATUS_REPORT-2018PAYMENT_STATUS_REPORT-2019PAYMENT_STATUS_REPORT-2020PAYMENT_STATUS_REPORT-2034PAYMENT_STATUS_REPORT-0518PAYMENT_STATUS_REPORT-0567PAYMENT_STATUS_REPORT-0573	This subscription has been cancelled because the bank details for its mandate are not valid.
bank	bank_account_closed	pad	PAYMENT_STATUS_REPORT-905	This subscription has been cancelled because the bank account for its mandate has been closed.
bank	direct_debit_not_enabled	pad	PAYMENT_STATUS_REPORT-907	This subscription has been cancelled because the bank account for its mandate does not support direct debit.
bank	bank_account_closed	pad	PAYMENT_STATUS_REPORT-910	This subscription was cancelled because the customer is deceased.
bank	authorisation_disputed	pad	PAYMENT_STATUS_REPORT-915	This subscription has been cancelled because the customer disputes authorising its mandate.
bank	other	pad	PAYMENT_STATUS_REPORT-914	An error was received from the banks while setting up the mandate for this subscription.
bank	refer_to_payer	pad	PAYMENT_STATUS_REPORT-914	An error was received from the banks while setting up the mandate for this subscription.
bank	mandate_cancelled	pad	PAYMENT_STATUS_REPORT-917PAYMENT_STATUS_REPORT-920	This subscription has been cancelled because its mandate was cancelled.
bank	mandate_cancelled	pa	<strong>Nous utilisons des cookies</strong>	
bank	mandate_cancelled	pa	Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.	
bank	mandate_cancelled	pa	Consultez notre politique de confidentialité	
bank	mandate_suspended_by_payer	pa		
bank	mandate_cancelled	pa		

Origin	Cause	Scheme	Reason code	Description
gocardless	mandate_cancelled	faster_payments	MERCHANT_MANDATE_CANCELLED	The mandate for this subscription was cancelled at your request.
gocardless	mandate_cancelled	faster_payments	MANDATE_EXPIRED	The subscription was cancelled because its mandate has expired.
gocardless	initial_one_off_payment_failed	-	-	This subscription has been cancelled because its mandate was cancelled.

**finished**

This subscription has finished. No further payments will be created.

Origin	Cause	Description
gocardless	subscription_finished	The subscription has finished.

**paused**

This subscription has been paused.

Origin	Cause	Description
api	subscription_paused	The subscription has been paused.
gocardless	subscription_paused	The subscription has been paused.

**resumed**

This subscription was resumed.

Origin	Cause	Description
api	subscription_resumed	The subscription was resumed.
gocardless	subscription_resumed	The subscription was resumed.

**scheduled\_pause**

This subscription has been scheduled to be paused at a future date.

Origin	Cause	Description
api	scheduled_pause	The subscription has been scheduled to be paused at a future date.

**scheduled\_pause\_cancelled**

An upcoming pause for this subscription has been cancelled.

Origin	Cause	Description
api	scheduled_pause_cancelled	An upcoming pause for this subscription has been cancelled.

**scheduled\_resume**

This paused subscription has been scheduled to be resumed at a future date.

Origin	Cause	Description
api	scheduled_resume	This paused subscription has been scheduled to be resumed at a future date.



HTTP

HTTP

```
GET https://api.gocardless.com/events/EV123 HTTP/1.1
```

```
HTTP/1.1 200 (OK)
Content-Type: application/json
```

```
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "subscriptions",
      "action": "payment_created",
      "links": [
        {
          "subscription": "SB123",
          "payment": "PM123"
        }
      ],
      "details": {
        "origin": "api",
        "cause": "payment_created",
        "description": "Payment created"
      },
      "metadata": {},
      "resource_metadata": {
        "order_dispatch_date": ""
      }
    }
  ]
}
```

**Nous utilisons des cookies**

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

**Success+**

## Intelligent Retries

Intelligent retries is a feature which automatically retries payments if they fail.

**Important:** To be able to use intelligent retries, Success+ needs to be enabled in [Gocardless Dashboard](#).

## Enabling Intelligent Retries

### Individual Payments

#### Creating Payment

When creating a [payment](#) set `retry_if_possible` to `true`. This will enable intelligent retries if the payment is considered eligible.

#### Updating Payment

If a payment has been created with `retry_if_possible` set to `true`, then it can be [updated](#) to `false`. This will prevent the payment being automatically retried in the event of a failure.

The API currently only allows the flag to be updated to `false`. Hence for a payment which has been already created it is not allowed to set this flag to `true`.



HTTP  
HTTP

```
POST https://api.gocardless.com/payments HTTP/1.1
Content-Type: application/json
{
  "payments": {
    "amount": 100,
    "currency": "GBP",
    "retry_if_possible": true,
    ...
  }
}

PUT https://api.gocardless.com/payments/PM123 HTTP/1.1
Content-Type: application/json
{
  "payments": {
    "retry_if_possible": false
  }
}
```

### Instalment Schedules

When creating an [instalment schedule](#) set `retry_if_possible` to `true` to ensure that all instalment payments created are automatically retried upon failure.



HTTP  
HTTP

```
POST https://api.gocardless.com/instalment_schedules HTTP/1.1
Content-Type: application/json
{
  "instalment_schedules": {
    "name": "Bike Invoice 271",
    "total_amount": "2500",
    "currency": "GBP",
    "retry_if_possible": true,
    ...
  }
}
```

### Subscriptions

When creating a [subscription](#) set `retry_if_possible` to `true` to ensure that all payments created for the subscription are retried automatically upon failure.



HTTP  
HTTP

```
POST https://api.gocardless.com/subscriptions HTTP/1.1
Content-Type: application/json
{
  "subscriptions": {
    "amount": "2500",
    "currency": "GBP",
    "interval_unit": "monthly",
    "retry_if_possible": true,
    ...
  }
}
```

## Handling Failures

On payment failure the failed [event](#)

When a payment is automatically

When `will_attempt_retry` is `true`,  
can lead to them being double charged.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

will be true.



Consultez notre politique de confidentialité

HTTP  
HTTP

```
GET https://api.gocardless.com/events/EV123 HTTP/1.1
HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-04-08T17:01:06.000Z",
      "resource_type": "payments",
      "action": "failed",
      "details": {
        "origin": "bank",
        "cause": "insufficient_funds",
        "description": "The customer's account had insufficient funds to make this payment.",
        "scheme": "sepa_core",
        "reason_code": "AM04",
        "will_attempt_retry": true
      },
      "metadata": {},
      "links": {
        "payment": "PM123"
      }
    }
  ]
}
```

## Retry Rules

The criteria below are subject to change and should not be relied upon. Use the `will_attempt_retry` field [above](#) to know for sure if a payment will be retried.

The `retry_if_possible` field does not guarantee the payment will be retried on failure. A payment will be retried if the following conditions are met:

- Intelligent retries for the scheme has been enabled on the [settings page](#).
- The payment has field `retry_if_possible` set to true.
- The payment has failed due to insufficient funds.
- The number of payment failures is lesser than the number of retries configured in the intelligent retries settings page.
- The subsequent retry can be confirmed within the retry window configured in the intelligent retries settings page.

## Helper Endpoints

### Healthcheck

We expose a health check endpoint which can be used to test connections to our API. Requests to this endpoint do not require authorization and are not rate limited. This endpoint will return a `200` response while our API is available.

- [https://api.gocardless.com/health\\_check](https://api.gocardless.com/health_check) for live
- [https://api-sandbox.gocardless.com/health\\_check](https://api-sandbox.gocardless.com/health_check) for sandbox

### Bank Details Lookups

Look up the name and reachability of a bank account.

#### Properties

##### available\_debit\_schemes

Array of [schemes](#) supported for this bank account. This will be an empty array if the bank account is not reachable by any schemes.

##### bank\_name

The name of the bank with which the account is held (if available).

##### bic

ISO 9362 SWIFT BIC of the bank with which the account is held.

Even if no BIC is returned for an account, GoCardless may still be able to collect payments from it - you should refer to the `available_debit_schemes` attribute to determine reachability.

### Perform a bank details lookup

Performs a bank details lookup. As part of the lookup, a modulus check and reachability check are performed.

For UK-based bank accounts, where an account holder name is provided (and an account number, a sort code or an iban are already present), we verify that the account holder name and bank account number match the details held by the relevant bank.

If your request returns an [error](#) or [warning](#), GoCardless may be able to collect payments from the specified bank account.

Bank account details may be supplied via the [ACH scheme](#) for compliance reasons.

If a bank account is discovered to be invalid during the flow, it will be rejected.

*Note:* Usage of this endpoint is managed by the [Bank Details Lookups](#) endpoint. Please stay in touch.

Relative endpoint: POST /bank\_details

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

from the specified bank account.

account can accept Direct Debit. To proceed in this check to continue with the payment collection, please get

#### Parameters

**account\_holder\_name**

The account holder name associated with the account number (if available). If provided and the country code is GB, a payer name verification will be performed.

**account\_number**

Bank account number - see [local details](#) for more information. Alternatively you can provide an [iban](#).

**bank\_code**

Bank code - see [local details](#) for more information. Alternatively you can provide an [iban](#).

**branch\_code**

Branch code - see [local details](#) for more information. Alternatively you can provide an [iban](#).

**country\_code**

[ISO 3166-1](#) alpha-2 code. Must be provided if specifying local details.

**iban**

International Bank Account Number. Alternatively you can provide [local details](#).

HTTP PHP Python Ruby Java JavaScript .NET Go

HTTP

```
POST https://api.gocardless.com/bank_details_lookups HTTP/1.1
Content-Type: application/json
{
  "bank_details_lookups": {
    "account_number": "55779911",
    "branch_code": "200000",
    "country_code": "GB"
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "bank_details_lookups": {
    "available_debit_schemes": ["bacs"],
    "bank_name": "BARCLAYS BANK PLC",
    "bic": "BARCGB22XXX"
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);

$client->bankDetailsLookups()->create([
  'params' => [
    "account_number" => "55779911",
    "branch_code" => "200000",
    "country_code" => "GB"
  ]
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.bank_details_lookups.create(params={
  "account_number": "55779911",
  "branch_code": "200000",
  "country_code": "GB"
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)

@client.bank_details_lookups.create(
  params: {
    country_code: "GB",
    account_number: "55779911",
    branch_code: "200000"
  }
)
```

Java

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();
```

```
BankDetailsLookup bankDetailsLookup =
  .withAccountNumber("55779911")
  .withBranchCode("200000")
  .withCountryCode("GB")
  .execute();
```

JavaScript

```
const constants = require('gocardless');
const gocardless = require('gocardless');
const client = gocardless('your_access_token');

const bankDetailsLookup = await client.bankDetailsLookups.create({
  account_number: "55779911",
  branch_code: "200000",
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        country_code: "GB"
    });
.NET

String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var bankDetailsRequest = new GoCardless.Services.BankDetailsLookupCreateRequest()
{
    AccountNumber = "55779911",
    BranchCode = "200000",
    CountryCode = "GB"
};

var bankDetailsResponse = await gocardless.BankDetailsLookups.CreateAsync(bankDetailsRequest);
GoCardless.Resources.BankDetailsLookup bankDetails = bankDetailsResponse.BankDetailsLookup;

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    bankDetailsLookupCreateParams := gocardless.BankDetailsLookupCreateParams{
        AccountNumber: "55779911",
        BranchCode:     "200000",
        CountryCode:   "GB",
    }

    bankDetailsLookup, err := client.BankDetailsLookups.Create(context, bankAuthorisationCreateParams)
}

```

## Mandate PDFs

Mandate PDFs allow you to easily display [scheme-rules compliant](#) Direct Debit mandates to your customers.

### Properties

#### `expires_at`

The date and time at which the `url` will expire (10 minutes after the original request).

#### `url`

The URL at which this mandate PDF can be viewed until it expires at the date and time specified by `expires_at`. You should not store this URL or rely on its structure remaining the same.

## Create a mandate PDF

Generates a PDF mandate and returns its temporary URL.

Customer and bank account details can be left blank (for a blank mandate), provided manually, or inferred from the ID of an existing [mandate](#).

By default, we'll generate PDF mandates in English.

To generate a PDF mandate in another language, set the `Accept-Language` header when creating the PDF mandate to the relevant [ISO 639-1](#) language code supported for the scheme.

### Scheme      Supported languages

ACH	English (en)
Autogiro	English (en), Swedish (sv)

Bacs	English (en)
BECS	English (en)

BECS NZ	English (en)
Betalingservice	Danish (da), English (en)

PAD	English (en)
SEPA Core	Danish (da), Du

Relative endpoint: POST /mandate

### Parameters

#### `account_holder_name`

Name of the account holder  
Name of the account holder

#### `account_number`

Bank account number - see

#### `account_type`

Bank account type. Required information.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

(sv)

8 characters.

Consultez notre politique de confidentialité

s. See [local details](#) for more

**address\_line1**  
The first line of the customer's address.

**address\_line2**  
The second line of the customer's address.

**address\_line3**  
The third line of the customer's address.

**bank\_code**  
Bank code - see [local details](#) for more information. Alternatively you can provide an `iban`.

**bic**  
SWIFT BIC. Will be derived automatically if a valid `iban` or [local details](#) are provided.

**branch\_code**  
Branch code - see [local details](#) for more information. Alternatively you can provide an `iban`.

**city**  
The city of the customer's address.

**company\_name**  
The customer's company name. Used to populate the "Customer Name or Company name" field on the PDF.

**country\_code**  
[ISO 3166-1](#) alpha-2 code. Required if providing local details.

**danish\_identity\_number**  
For Danish customers only. The civic/company number (CPR or CVR) of the customer. Should only be supplied for Betalingsservice mandates.

**family\_name**  
The customer's family name (i.e. last name). Used to populate the "Customer Name or Company name" field on the PDF. Ignored if `company_name` is provided.

**given\_name**  
The customer's given name (i.e. first name). Used to populate the "Customer Name or Company name" field on the PDF. Ignored if `company_name` is provided.

**iban**  
International Bank Account Number. Alternatively you can provide [local details](#). IBANs cannot be provided for Autogiro mandates.

**mandate\_reference**  
Unique 6 to 18 character reference. This may be left blank at the point of signing.

**payer\_ip\_address**  
For American customers only. IP address of the computer used by the customer to set up the mandate. This is required in order to create compliant Mandate PDFs according to the ACH scheme rules.

**phone\_number**  
The customer phone number. Should only be provided for BECS NZ mandates.

**postal\_code**  
The customer's postal code.

**region**  
The customer's address region, county or department. For US customers a 2 letter [ISO3166-2:US](#) state code is required (e.g. CA for California).

**scheme**  
Direct Debit scheme. Can be supplied or automatically detected from the bank account details provided. If you do not provide a scheme, you must provide either a mandate, an `iban`, or [local details](#) including a `country_code`.

**signature\_date**  
If provided, a form will be generated with this date and no signature field.

**subscription\_amount**  
For American customers only. Subscription amount being authorised by the mandate. In the lowest denomination for the currency (cents in USD). Is required if `subscription_frequency` has been provided.

**subscription\_frequency**  
For American customers only. Frequency of the subscription being authorised by the mandate. One of `weekly`, `monthly` or `yearly`. Is required if `subscription_amount` has been provided.

**swedish\_identity\_number**  
For Swedish customers only. The civic/company number (personnummer, samordningsnummer, or organisationsnummer) of the customer. Should only be supplied for Autogiro mandates.

**links[creditor]**  
ID of an existing [creditor](#). Only required if your account manages multiple creditors.

**links[mandate]**  
ID of an existing [mandate](#) to build the PDF from. The customer's bank details will be censored in the generated PDF. No other parameters may be provided alongside this.

HTTP PHP Python Ruby Java JavaScript .NET Go  
HTTP

```
POST https://api.gocardless.com/mandate_pdfs HTTP/1.1
```

Content-Type: application/json

```
{
  "mandate_pdfs": {
    "links": {
      "mandate": "MD123"
    }
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "mandate_pdfs": {
    "url": "https://mandate-previews.gocardless.com/?token=1hu1xcPEbT9v3W00UbB0xh1GUSYEav004VVrp07YnUKRYvnP5",
    "expires_at": "2014-05-08T"
  }
}
```

POST https://api.gocardless.co

Content-Type: application/json

```
{
  "mandate_pdfs": {
    "account_number": "4477991",
    "branch_code": "200000",
    "country_code": "GB"
  }
}
```

HTTP/1.1 200 OK

Content-Type: application/json

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
"mandate_pdfs": {
  "url": "https://mandate-previews.gocardless.com/?token=vlaPBHzSvm1OPwDNatZYWJCM7XZcCUuLPn7m5XV5",
  "expires_at": "2014-05-08T17:01:06.000Z"
}
```

```
POST https://api.gocardless.com/mandate_pdfs HTTP/1.1
Content-Type: application/json
Accept-Language: fr
{
  "mandate_pdfs": {
    "iban": "FR14BARC20000055779911"
  }
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "mandate_pdfs": {
    "url": "https://mandate-previews.gocardless.com/?token=vlaPBHzSvm1OPwDNatZYWJCM7XZcCUuLPn7m5XV5",
    "expires_at": "2014-05-08T17:01:06.000Z"
  }
}
```

```
POST https://api.gocardless.com/mandate_pdfs HTTP/1.1
Content-Type: application/json
{
  "mandate_pdfs": {
    "account_number": "44779911",
    "branch_code": "200000",
    "country_code": "GB",
    "links": {
      "mandate": "MD123"
    }
  }
}
```

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json
{
  "error": {
    "message": "Cannot provide both a mandate and other details",
    "documentation_url": "https://developer.gocardless.com/api-reference#validation_failed",
    "type": "validation_failed",
    "request_id": "bd271b37-a2f5-47c8-b461-040dfe0e9cb1",
    "code": 422,
    "errors": [
      {
        "reason": "validation_failed",
        "message": "Cannot provide both a mandate and other details"
      }
    ]
  }
}
```

PHP

```
$client = new \GoCardlessPro\Client([
  'access_token' => 'your_access_token_here',
  'environment'  => \GoCardlessPro\Environment::SANDBOX
]);
```

```
$client->mandatePdfs()->create([
  'params' => ['links' => ['mandate' => "MD123"]]
]);
```

```
$client->mandate_pdfs()->create([
  'params' => ['account_number' => "44779911",
                "branch_code" => "200000",
                "country_code" => "GB"]
]);
```

```
$client->mandatePdfs()->create([
  'params' => ['iban' => "FR14BARC20000055779911"],
  'headers' => ['accept-language' => 'fr']
]);
```

Python

```
import gocardless_pro
client = gocardless_pro.Client(access_token="your_access_token_here", environment='sandbox')

client.mandate_pdfs.create(params={
  "links": {"mandate": "MD123"}
})

client.mandate_pdfs.create(params={
  "account_number": "44779911",
  "branch_code": "200000",
  "country_code": "GB"
})
```

```
client.mandate_pdfs.create(params={
  "iban": "FR14BARC20000055779911"
}, headers={
  "Accept-Language": "fr"
})
```

Ruby

```
@client = GoCardlessPro::Client.new(
  access_token: "your_access_token",
  environment: :sandbox
)
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
@client.mandate_pdfs.create(
  params: {
    links: { mandate: "MD123" }
  }
)

@client.mandate_pdfs.create(
  params: {
    account_number: "55779911",
    branch_code: "200000",
    country_code: "GB"
  }
)

@client.mandate_pdfs.create(
  params: {
    iban: "FR14BARC20000055779911",
  },
  headers: {
    "Accept-Language" => "fr"
  }
)
Java
```

```
import static com.gocardless.GoCardlessClient.Environment.SANDBOX;
String accessToken = "your_access_token_here";
GoCardlessClient client = GoCardlessClient
  .newBuilder(accessToken)
  .withEnvironment(SANDBOX)
  .build();

MandatePdf mandatePdfForMandate = client.mandatePdfs().create()
  .withLinksMandate("MD123")
  .execute();

MandatePdf mandatePdfForBankDetails = client.mandatePdfs.create()
  .withAccountNumber("44779911")
  .withBranchCode("200000")
  .withCountryCode("GB")
  .execute();

MandatePdf mandatePdfInFrench = client.mandatePdfs.create()
  .withIban("FR14BARC20000055779911")
  .withHeader("Accept-Language", "fr")
  .execute();
```

JavaScript

```
const constants = require('gocardless-nodejs/constants');
const gocardless = require('gocardless-nodejs');
const client = gocardless('your_access_token_here', constants.Environments.Sandbox);

await client.mandatePdfs.create({
  links: {
    mandate: "MD123"
  }
});

await client.mandatePdfs.create(
{
  links: {
    mandate: "MD123"
  }
},
"mandate_pdfs_idempotency_key"
);

await client.mandatePdfs.create(
{
  account_number: "44779911",
  branch_code: "200000",
  country_code: "GB"
},
"",
{ "Accept-Language": "fr" }
);
```

.NET

```
String accessToken = "your_access_token";
GoCardlessClient gocardless = GoCardlessClient.Create(accessToken, Environment.SANDBOX);

var mandatePdfRequest = new GoCardless.Services.MandatePdfsCreateRequest()
{
  AccountNumber = "55779911",
  BranchCode = "200000",
  CountryCode = "GB"
};

var mandatePdfResponse = await
GoCardless.Resources.MandatePdfsCreateAsync(mandatePdfRequest);

var mandatePdfRequest = new GoCardless.Services.MandatePdfsCreateRequest()
{
  Iban = "FR14BARC20000055779911"
};

var requestSettings = new GoCardless.Resources.MandatePdfsCreateRequestSettings()
{
  CustomiseRequestMessage = "Mandate PDF in French"
};

var mandatePdfResponse = await
GoCardless.Resources.MandatePdfsCreateAsync(mandatePdfRequest);
```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

Go

package main

import (
    gocardless "github.com/gocardless/gocardless-pro-go/v2"
)

func main() {
    accessToken := "your_access_token_here"
    config, err := gocardless.NewConfig(accessToken, gocardless.WithEndpoint(gocardless.SandboxEndpoint))
    if err != nil {
        fmt.Printf("got err in initialising config: %s", err.Error())
        return
    }
    client, err := gocardless.New(config)
    if err != nil {
        fmt.Println("error in initialising client: %s", err.Error())
        return
    }

    mandatePdfCreateParams := gocardless.MandatePdfCreateParams{
        Links: &gocardless.MandatePdfCreateParamsLinks{
            Mandate: "MD123",
        },
    }

    mandatePdf, err := client.MandatePdfs.Create(context, mandatePdfCreateParams)

    requestOption := gocardless.WithIdempotencyKey("mandate_pdfs_idempotency_key")
    mandatePdf, err = client.MandatePdfs.Create(context, mandatePdfCreateParams, requestOption)

    headers := map[string]string{"Accept-Language": "fr"}
    requestOption = gocardless.WithHeaders(headers)
    mandatePdf, err = client.MandatePdfs.Create(context, mandatePdfCreateParams, requestOption)
}

```

## Appendix

### Webhooks

Check out our getting started [guide](#) for a step-by-step introduction to handling webhooks, with PHP, Ruby, Python and Java code samples

#### Overview

A webhook is a HTTP request containing a list of GoCardless events. Webhooks notify you of new [events](#), e.g. when the bank informs us that one of your payments has failed.

You can enable webhooks by creating a `Webhook Endpoint` in your GoCardless dashboard.

When an event occurs in your GoCardless account, we'll batch that (and any other undelivered events) up into a single webhook and send it to every enabled webhook endpoint as a `POST` request.

There are a few things to note when using webhooks:

- Webhooks may arrive out of order, or the same one may be delivered multiple times. Each event may be included in one or more webhooks. We can only guarantee that we'll try to deliver each event at least once.
- When deciding what actions to take in response to Webhook events, we recommend you switch on the `details[cause]` field. Other fields, such as `details[reason_code]`, are payment scheme-specific and can be inconsistent between banks, whereas the `details[cause]` field is our simplified and predictable key indicating what triggered the event.
- Webhooks with an invalid signature must return a `498 Token Invalid` error.
- Webhooks about unknown events should be ignored, and return `204 No Content`. GoCardless may add new events to the API without considering this a backwards incompatible change.
- You must use `SSL/TLS` for webhook URLs. Unsecured webhook URLs are only allowed in the sandbox environment. Your server must respond with the full certificate chain including intermediate certificates. You can test this with a tool like [Qualys SSL labs](#)
- Webhooks include an `Origin` header indicating what GoCardless environment they were sent from. This will be <https://api.gocardless.com> for live, and <https://api-sandbox.gocardless.com> for sandbox.
- Webhooks you've received in the last 3 months are viewable in your GoCardless dashboard in the “Developers” area.
- The webhook ID included under the `meta` key should not be used for deduplication: it is only included to help with debugging. Searching for this ID in the GoCardless dashboard will show you the information on the most recent delivery attempt for this webhook.

Our webhooks platform will only execute any work that needs doing

#### Status codes

Your webhook handler should return `Invalid Token`

#### Errors & Retries

In the event we fail to deliver the intervals, including the initial attem

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

tore the webhook body and

nvalid, you should return a `498`

8 times at increasing time

You can view webhooks we've sent you in your GoCardless dashboard, and can retry them if required.

## Approved IP addresses

We send webhooks from the following IP addresses which you may wish to add to your firewalls approved list:

- 35.204.73.47
- 35.204.191.250
- 35.204.214.181

We will provide advance notification by email at least two weeks before we make any changes to these addresses.

You can set the email we will contact you at from your Dashboard - simply click "Settings" in the top-right hand corner, then "Contact preferences", and then edit your developer contact.

## Signing Webhooks

We sign the body of the POST request with an HMAC SHA256 hex digest, using the secret of the webhook endpoint for which this webhook is being sent. This is done using an additional header:

### Webhook-Signature

The HMAC SHA256 hex digest of the request body.

You must check that the webhook has a valid signature before processing it. Here's how you could do that in Ruby:

```
# request_signature - the signature sent in Webhook-Signature
#       request_body - the JSON body of the webhook request
#               secret - the secret for the webhook endpoint

require "openssl"

digest = OpenSSL::Digest.new("sha256")
calculated_signature = OpenSSL::HMAC.hexdigest(digest, secret, request_body)

if calculated_signature == request_signature
  # Signature ok!
else
  # Invalid signature. Ignore the webhook and return 498 Token Invalid
end
```

## Examples

### Payments

In this example, you have set up two customers, and each has a mandate. You created a payment for each customer, called "Test Payment" and "Test Payment 2" respectively. A few days later, these have been collected, and this has generated a payment update event for those payments. You have created a webhook endpoint, with id WE123, secret 123ABC456DEF, and url <https://example.com/webhooks>. To process it, you should:

1. Store the contents of the webhook in durable storage e.g a database or queue.
2. Check the signature
3. Return 204 No Content
4. Process the content of the webhook asynchronously by
  1. Checking that you have not already processed the events contained in the webhook.
  2. Fetching the updated resources, using the ID's supplied, and check that they have not changed further since the webhook was sent (since webhooks may arrive out of order)
  3. Acting on the events, e.g. shipping goods, extending subscription



HTTP

HTTP

```
POST https://example.com/webhooks HTTP/1.1
User-Agent: gocardless-webhook-service/1.1
Origin: https://api-sandbox.gocardless.com
Content-Type: application/json
Webhook-Signature: 8d625e2359a8a2f9161e0ec9a534a9f75c47aa7b305b18685929ad58f08d2efc
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-08-03T12:00:00.000Z",
      "action": "confirmed",
      "resource_type": "payments",
      "links": {
        "payment": "PM123"
      },
      "details": {
        "origin": "gocardless",
        "cause": "payment_confirmed",
        "description": "Payment confirmed"
      }
    },
    {
      "id": "EV456",
      "created_at": "2014-08-03T12:00:00.000Z",
      "action": "failed",
      "resource_type": "payments",
      "links": {
        "payment": "PM456"
      },
      "details": {
        "origin": "bank",
        "cause": "payment_failed"
      }
    }
  ]
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

        "cause": "mandate_cancelled",
        "description": "Customer cancelled the mandate at their bank branch.",
        "scheme": "bacs",
        "reason_code": "ARUDD-1"
    }
},
"meta": { "webhook_id": "WB123" }
}

```

## Mandates

In this example a customer has two mandates, and has just closed their bank account. Both mandates were cancelled by their bank, and this is being passed on through GoCardless via a webhook, using the same webhook endpoint as in the previous example. Once again, you should:

1. Store the contents of the webhook in durable storage e.g a database or queue.
2. Check the signature
3. Return 204 No Content
4. Process the content of the webhook asynchronously by
  1. Checking that you have not already processed the events contained in the webhook.
  2. Fetching the updated resources, using the ID's supplied, and check that they have not changed further since the webhook was sent (since webhooks may arrive out of order)
  3. Acting on the events, e.g. shipping goods, extending subscription



HTTP  
HTTP

```

POST https://example.com/webhooks HTTP/1.1
User-Agent: gocardless-webhook-service/1.1
Origin: https://api-sandbox.gocardless.com
Content-Type: application/json
Webhook-Signature: c1ec421b5f7a1f06172ee1c50d013fe9a91f0db899718f88f9d4a32a4e2180e5
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-08-04T12:00:00.000Z",
      "action": "cancelled",
      "resource_type": "mandates",
      "links": {
        "mandate": "MD123"
      },
      "details": {
        "origin": "bank",
        "cause": "bank_account_disabled",
        "description": "Your customer closed their bank account.",
        "scheme": "bacs",
        "reason_code": "ADDACS-B"
      }
    },
    {
      "id": "EV456",
      "created_at": "2014-08-04T12:00:00.000Z",
      "action": "expired",
      "resource_type": "mandates",
      "links": {
        "mandate": "MD456"
      },
      "details": {
        "origin": "gocardless",
        "cause": "mandate_expired",
        "description": "The mandate expired due to inactivity."
      }
    }
  ],
  "meta": { "webhook_id": "WB123" }
}

```

## Payouts

In this example, your payments have been collected by GoCardless, and are now ready to be paid out into the associated [creditor bank account](#). This payout has just been generated, which triggers a webhook using the same webhook endpoint as in the previous examples. Once again you should:

1. Store the contents of the webhook in durable storage e.g a database or queue.
2. Check the signature
3. Return 204 No Content
4. Process the content of the webhook asynchronously by
  1. Checking that you have not already processed the events contained in the webhook.
  2. Fetching the updated resources, using the ID's supplied, and check that they have not changed further since the webhook was sent (since webhooks may arrive out of order)
  3. Acting on the events,



HTTP  
HTTP

```

POST https://example.com/webhooks HTTP/1.1
User-Agent: gocardless-webhook-service/1.1
Origin: https://api-sandbox.gocardless.com
Content-Type: application/json
Webhook-Signature: 42aa2860ecb1
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-08-04T12:00:00.000Z",
      "action": "paid_out"
    }
  ],
  "meta": { "webhook_id": "WB123" }
}

```

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "action": "paid",
    "resource_type": "payouts",
    "links": {
      "payout": "P0123"
    }
  ],
  "meta": { "webhook_id": "WB123" }
}

```

## Subscriptions

In this example a subscription has just raised a payment, which triggers a webhook using the same webhook endpoint as in the previous examples. Once again you should:

1. Store the contents of the webhook in durable storage e.g a database or queue.
2. Check the signature
3. Return 204 No Content
4. Process the content of the webhook asynchronously by
  1. Checking that you have not already processed the events contained in the webhook.
  2. Fetching the updated resources, using the ID's supplied, and check that they have not changed further since the webhook was sent (since webhooks may arrive out of order)
  3. Acting on the events, e.g. shipping goods, extending subscription



HTTP  
HTTP

```

POST https://example.com/webhooks HTTP/1.1
User-Agent: gocardless-webhook-service/1.1
Origin: https://api-sandbox.gocardless.com
Content-Type: application/json
Webhook-Signature: da7db0aa5e63b77b3890e7ca85c129a8b317f7605f7d6e43466751a70ff4afe7
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-08-04T12:00:00.000Z",
      "action": "payment_created",
      "resource_type": "subscriptions",
      "links": {
        "subscription": "SB123",
        "payment": "PM123"
      }
    }
  ],
  "meta": { "webhook_id": "WB123" }
}

```

## JavaScript Flow

You can use our JavaScript flow to create custom payment pages. Our JavaScript library will send sensitive bank details to our servers directly and return a token that can be used to create a [customer bank account](#).

The API allows you to create and show your customer bank account tokens.

**Restricted:** the JavaScript Flow is [restricted](#) to [GoCardless Pro & Enterprise](#) accounts with approved payment pages. To instead use the GoCardless hosted payment pages, see the [redirect flows](#) endpoint.

## Customer Bank Account Tokens

Customer bank account tokens can only be used once. Attempting to create a second customer bank account from the same customer bank account token will result in a `customer_bank_account_token_used` error.

### Properties

<code>id</code>	Unique identifier, beginning with “AT”.
<code>created_at</code>	Fixed <a href="#">timestamp</a> of when the customer bank account token was originally created.
<code>links[customer_bank_account]</code>	ID of <a href="#">customer bank account</a> that was created from this token. <i>Note:</i> this property will only be present if this token has been used to create a customer bank account.

### Create a customer bank account token

Creates a new customer bank account

Relative endpoint: POST /customers/:customer\_id/bacs

**Note:** The customer bank account token we may not honour the usual [bac](#)

GoCardless performs modulus checks on the bank account numbers it can use.

The overall flow is:

1. You render a payment form

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

e specific set of bank details you



Consultez notre politique de confidentialité

2. Your customer supplies their bank account details and submits the form. The form submission needs to be intercepted in JavaScript and submitted using our JavaScript library; on success you should inject the created token into the form and complete the submission to your server.
3. You create a [customer bank account](#) passing in the token in place of bank account details.

## Intercepting form submission with the JavaScript library

### Parameters

Bank account details may either be supplied using the IBAN (international bank account number), or [local details](#). For more information on the different fields required in each country, please see the [local bank details](#) section.

```
publishable_access_token
  required can only be sent as a top level parameter. This token will be sent through in the Authorization header when the request is made.
account_number
  Bank account number.
bank_code
  Bank identifying code.
branch_code
  Branch identifying code.
iban
  Valid international bank account number.
account_type
  Bank account type. Only required for USD denominated bank accounts - see local details for more information.
country_code
  ISO 3166-1 alpha-2 code. Defaults to the country code of the iban if supplied, otherwise is required.
currency
  ISO 4217 currency code, defaults to national currency of country_code.
account_holder_name
  required Name of the account holder, as it appears with the bank. This may not be more than 18 characters.
```

```
<!DOCTYPE html>
<html>
  <body>
    <script src="https://pay-sandbox.gocardless.com/js/beta"></script>
    <!-- For the live environment, use https://pay.gocardless.com/js/beta -->
    <form id="form"
      action="/complete"
      method="post"
      onsubmit="onSubmit(event)">
      <input name="publishable_access_token" id="publishable_access_token" type="hidden"
        value="JF00001078YEW7HHA0RA4Z73J7PYBN">
      <input name="given_name" id="given_name" type="text">
      <input name="family_name" id="family_name" type="text">
      <input name="address_line1" id="address_line1" type="text">
      <input name="city" id="city" type="text">
      <input name="region" id="region" type="text">
      <input name="postal_code" id="postal_code" type="text">
      <input name="country_code" id="country_code" type="text">
      <input id="account_number" type="text">
      <input id="account_type" type="text">
      <input id="bank_code" type="text">
      <input id="branch_code" type="text">
      <input id="iban" type="text">
      <input id="account_holder_name" type="text">
      <input id="bank_accounts_country_code" type="text">
      <input type="submit" value="Continue">
      <h1 id="error"></h1>
    </form>

    <script>
      function onSubmit(event) {
        var form = event.target;
        var publishableAccessToken = document.getElementById('publishable_access_token').value;

        GoCardless.customerBankAccountTokens.create({
          publishable_access_token: publishableAccessToken,
          customer_bank_account_tokens: {
            account_number: document.getElementById('account_number').value,
            bank_code: document.getElementById('bank_code').value,
            branch_code: document.getElementById('branch_code').value,
            iban: document.getElementById('iban').value,
            account_holder_name: document.getElementById('account_holder_name').value,
            country_code: document.getElementById('bank_accounts_country_code').value
          }
        }, function(response) {
          if (response.error) {
            document.getElementById('error')
              .textContent = 'Error: ' + response.error.message;
          } else {
            var input = document.createElement('input');
            input.type = 'hidden';
            input.value = response.id;
            input.name = 'customer_bank_account_token';
            form.appendChild(input);

            form.submit();
          }
        });
        // Prevent form submission
        event.preventDefault();
      }
    </script>
  </body>
</html>
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

Creating a [customer bank account](#) after the form is submitted.

```
require "gocardless_pro"
require "sinatra"

post "/complete" do

  client = GoCardlessPro::Client.new(access_token: "access_token")
  # You can also pass in environment as :sandbox to make requests to the sandbox environment rather than production

  customer_params = {
    email: params[:email],
    given_name: params[:given_name],
    family_name: params[:family_name],
    address_line1: params[:address_line1],
    city: params[:city],
    postal_code: params[:postal_code],
    country_code: params[:country_code],
  }

  customer = client.customers.create(params: customer_params)

  customer_bank_account_params = {
    links: {
      customer: customer.id,
      customer_bank_account_token: params[:customer_bank_account_token]
    }
  }

  customer_bank_account = client.customer_bank_accounts.create(
    params: customer_bank_account_params
  )

  new_mandate = client.mandates.create(params: {
    links: {
      customer_bank_account: customer_bank_account.id
    }
  })

  puts new_mandate.inspect

  "Thanks!"
end
```

## Get a single customer bank account token

Retrieves the details of an existing customer bank account token.

Relative endpoint: GET /customer\_bank\_account\_tokens/AT123

HTTP  
HTTP

GET https://api.gocardless.com/customer\_bank\_account\_tokens/AT123 HTTP/1.1

HTTP/1.1 200 (OK)  
Content-Type: application/json  
{  
 "customer\_bank\_account\_tokens": {  
 "id": "AT123",  
 "created\_at": "2014-05-08T17:01:06.000Z",  
 "links": {  
 "customer\_bank\_account": "BA123"  
 }  
 }  
}

## OAuth

Check out our getting started [guide](#) for a step-by-step introduction to building an OAuth integration, with PHP, Ruby, Python and Java code samples

OAuth allows you to work with other users' GoCardless accounts. Once a user approves you, you can use the GoCardless API on their behalf and receive their [webhooks](#).

The GoCardless OAuth API conforms to the [OAuth spec](#), and can be used with OAuth client libraries [available in most languages](#).

The base URLs for the GoCardless OAuth API are:

- <https://connect.gocardless.com> for live
- <https://connect-sandbox.gocardless.com> for sandbox

### The OAuth flow

The flow to link a GoCardless account:

1. The user clicks a link on your site that wants the user to authorise.
2. On the GoCardless OAuth page, the user grants permission to your app.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

This defines the level of access you have to the user's account.

Grant access to my account to your app.



Consultez notre politique de confidentialité



# GoCardless

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

1. The user, having agreed that they'd like to connect to your app, is redirected back to your site with a `code`.
2. Your site exchanges the `code` for a permanent `access_token` which can be stored and used to make requests to the API on the user's behalf.
3. You can then make API requests using the user's access token and receive webhooks relating to their account.

You can try the process for yourself by connecting your sandbox account to our [example app](#).

## Creating an OAuth app

To get started, you'll need a GoCardless account. If you haven't already, you can sign up [here](#) to get one. Once you're logged into your account, you can create apps from the [developer section of your dashboard](#)

When you do this, you'll be issued a `client_id` and `client_secret`, each 64 characters long, which you'll use to identify your integration when requesting access to other users' GoCardless accounts.

## Building an authorisation link

An authorisation link takes your user to the GoCardless OAuth flow, where they may agree to connect their GoCardless account to your app. It includes details of your app and the level of access requested.

To construct an authorisation link, take the [relevant OAuth base URL](#) plus the relative endpoint `/oauth/authorize` with the following querystring parameters. You should redirect your user to this link using a `GET` request.

### Parameters

`response_type`  
*required* The kind of OAuth request you're making - only `code` is supported.

`client_id`  
*required* The `client_id` of your app.

`scope`  
*required* The level of access you want to your users' GoCardless accounts - this may either be `read_write` or `read_only`.

`redirect_uri`  
*required* The URL to send your users to once they've agreed to connect their account to GoCardless (as well if they deny authorisation or something goes wrong, in which case we'll pass you details of the error). This must exactly match one of the `redirect_uris` stored on your app.

`state`  
An optional string of your choice. Any value you pass in here will be included as a querystring parameter when we redirect back to your redirect URL. Please note that this value can be tampered with by your user, and so shouldn't be trusted implicitly. We recommend using this parameter for a [CSRF token](#).

`prefill[email]`  
Your user's email address. We will pre-fill this on the login or signup forms to make it quicker and easier for them to complete the flow.

`prefill[given_name]`  
Your user's given (first) name. We will pre-fill this on the signup form.

`prefill[family_name]`  
Your user's family (last) name. We will pre-fill this on the signup form.

`prefill[organisation_name]`  
The name of the user's organisation (e.g. Acme Widgets Ltd, London Scout Group or Tim Rogers). We will pre-fill this on the signup form.

`prefill[country_code]`  
The country code of the user

`language`  
The language that the login view is in, to the most appropriate available

`initial_view`  
An optional parameter, set to the login view if you're requesting



HTTP  
HTTP

GET [https://connect.gocardless.com/api/auth/authorize?client\\_id=...&response\\_type=code&scope=read\\_write&redirect\\_uri=https://...](https://connect.gocardless.com/api/auth/authorize?client_id=...&response_type=code&scope=read_write&redirect_uri=https://...)

## Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

it. Otherwise, we will fall back

low. By default, they will see the

```
client_id=sx6WHUAVMUrinkJNJn8DotVFm&
scope=read_write&
redirect_uri=http%3A%2Flocalhost%2Fcallback&
state=q8wEr9yMohTP&
prefill[email]=tim%40gocardless.com,
prefill[given_name]=Tim,
prefill[family_name]=Rogers,
prefill[organisation_name]=Tim%27s%20Fishing%20Store
prefill[country_code]=GB
```

## Handling the redirect back to you

Once the user has connected their GoCardless account to your app, they'll be redirected to your `redirect_uri`.

In the querystring, you will be passed a code, which you must exchange for a long-lived access token within 5 minutes. If you provided a `state` in the original redirect to GoCardless, this will be passed back to you.

## Errors

If the user denies your request to connect, or some other kind of error occurs (for example you specify an invalid `scope` or `response_type` is missing), the user will be redirected back to you with an `error` and `error_description`, per the [OAuth spec](#).

`error` will be set to one of the following, and there will be a human-readable `error_description` which you may wish to store to refer to in case of errors:

```
invalid_request
    You failed to provide either a scope or a response_type.
invalid_scope
    You provided a scope other than read_only or read_write.
unsupported_response_type
    You provided a response_type other than code.
access_denied
    The user chose not to connect their account to your app.
```



HTTP  
HTTP

```
GET https://localhost/callback HTTP/1.1
code=6NjigQzTHeGEGsAZXUmaBfB&
state=q8wEr9yMohTP

GET https://localhost/callback HTTP/1.1
error=access_denied&
error_description=The%20user%20cancelled%20the%20authorisation%20process.
```

## Exchanging an authorisation code for an access token

Exchanges the authorisation code passed to you in the redirect for an access token which you may store and use to make future requests on the user's behalf. Note that if the user has previously authorised with you, we will still generate a new access token and will disable any existing tokens for that user.

Relative endpoint: POST /oauth/access\_token

### Parameters

`grant_type` *required* The kind of OAuth grant you've received - this will be `authorization_code`.  
`code` *required* The authorisation code that resulted from the user agreeing to connect their account to your app, passed to you in the querystring as `code`. This will expire after 5 minutes, so you should exchange it for an access token immediately.  
`redirect_uri` *required* One of the `redirect_uris` set on your app, and which you provided when sending your user to the OAuth flow.  
`client_id` *required* The `client_id` for your registered app.  
`client_secret` *required* The `client_secret` for your registered app.

### Response

Following the [OAuth spec](#), this endpoint responds with JSON including an `access_token` if successful.

`access_token` Your permanent access token for authenticating on behalf of the user. You should store this for future use.  
`scope` The scope of the access token, as originally requested (either `read_write` or `read_only`).  
`token_type` The type of token you've been given.  
`organisation_id` The ID of the GoCardless account that a [webhook](#) relates to.  
`email` A contact email for the GoCardless account.

### Errors

In case of an error, the endpoint returns:

`invalid_request` You have not provided at least one of the required parameters, or one of your parameters is of the wrong type.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

to identify to which of your users

one of the following:



Consultez notre politique de confidentialité

, or one of your parameters is of the wrong type.

```

unsupported_grant_type
    You provided a grant_type other than authorization_code.

invalid_client
    The client_id and client_secret you provided to authenticate as your app were invalid or refer to a disabled app.

invalid_grant
    The code you provided has already been used, has expired, or the redirect_uri you provided does not match the one originally provided when passing the user to GoCardless.

HTTP
HTTP

POST https://connect.gocardless.com/oauth/access_token HTTP/1.1
grant_type=authorization_code&
code=6NjigQxzTHcgEGsAZXUmaBfB&
redirect_uri=http%3A%2Flocalhost%2Fcallback&
client_id=sx6WHAUAVMuinkJNjn8dotFm&
client_secret=exaserfrWrPdxADDUBWVqGbPbF

HTTP/1.1 200 (OK)
Content-Type: application/json
{
  "access_token": "e72e16c7e42f292c6912e7710c123347ae178b4a",
  "scope": "read_write",
  "token_type": "bearer",
  "email": "accounts@acme.com",
  "organisation_id": "OR123"
}

POST https://connect.gocardless.com/oauth/access_token HTTP/1.1
grant_type=authorization_code&
code=goQcmY8MebbAuKy8L&
redirect_uri=http%3A%2Flocalhost%2Fcallback&
client_id=sx6WHAUAVMuinkJNjn8dotFm&
client_secret=exaserfrWrPdxADDUBWVqGbPbF

HTTP/1.1 400 (Bad Request)
Content-Type: application/json
{
  "error": "invalid_grant",
  "error_description": "code has already been used"
}

```

## Making requests

You can now make standard [API requests](#) using the access token you received in exchange for the authorisation code. This will allow you to view and manipulate (depending on the scope you requested) the user's data.

As with [elsewhere](#) in the API, the access token must be provided in an `Authorization` request header, using the [bearer](#) authentication scheme.

`Authorization: Bearer e72e16c7e42f292c6912e7710c123347ae178b4a`

Any errors generated by requests using your users' access tokens will follow the [usual structure](#) of errors in the GoCardless API, as opposed to the OAuth-specific errors above.

If the access token provided is invalid, you will receive a `401 Unauthorized` response. The response body will include a reason, which will be either:

- `access_token_not_found` (the access token was not recognised); or
- `access_token_revoked` (the user has revoked your access to their account); or
- `access_token_not_active` (the access token is inactive for another reason)

Users will need to go through the [authorisation flow](#) again in order for you to obtain a new access token.

If you receive an unexpected `403 Forbidden` response for an endpoint your app is authorised to access (see below), this may indicate that your user's account is under review. The user will have been informed by GoCardless that they need to provide additional information to re-enable their account.

**Restricted:** You will not be able to access the following API endpoints on behalf of organisations connected to your app:

- Creditor (create and update)
- Creditor Bank Account (all endpoints)

Unless your app's payment pages that have been approved as [scheme rules compliant](#) by our sponsor bank you must use GoCardless' hosted payment pages (via the [Redirect Flow API](#)). As such, the following endpoints will also be restricted by default:

- Mandate (create and reinstate)
- Customer (create)
- Customer bank account (create)



HTTP  
HTTP

POST https://api.gocardless.com/mandates HTTP/1.1  
Authorization: Bearer e72e16c7e42f292c6912e7710c123347ae178b4a

HTTP/1.1 200 (OK)

GET https://api.gocardless.com/mandates/{id} HTTP/1.1  
Authorization: Bearer access\_token

HTTP/1.1 401 (Unauthorized)  
Content-Type: application/json

```
{
  "error": {
    "code": 401,
    "type": "invalid_api_usage",
    "errors": [
      ...
    ]
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```
{
  "message": "The access token you provided has been revoked",
  "reason": "access_token_revoked"
},
"documentation_url": "https://developer.gocardless.com/api-reference/#access_token_revoked",
"message": "The access token you provided has been revoked",
"request_id": "dd50eaaf-8213-48fe-90d6-5466872efbc4"
}
```

## Receiving webhooks

If you've provided a webhook URL for your app, we'll notify you of new [events](#) relating to the GoCardless accounts of your users. For details on how webhooks work and how to process them, see the [guide](#).

The events we send you will be identical to those described, but with an additional `links[organisation]` attribute allowing you to identify which of your users a webhook relates to.

The organisation ID is provided when [exchanging an authorisation code for an access token](#) and when [looking up an access token](#). If you expect to handle webhooks, you should store it for future reference.



HTTP  
HTTP

```
POST https://example.com/webhooks HTTP/1.1
User-Agent: gocardless-webhook-service/1.1
Content-Type: application/json
Webhook-Signature: 86f8bb84a4de63cff4af48f22b64b20970b415b066e3d21459ea515052860514
{
  "events": [
    {
      "id": "EV123",
      "created_at": "2014-08-03T12:00:00.000Z",
      "action": "confirmed",
      "resource_type": "payments",
      "links": {
        "payment": "PM123",
        "organisation": "OR123"
      },
      "details": {
        "origin": "gocardless",
        "cause": "payment_confirmed",
        "description": "Payment was confirmed as collected"
      }
    },
    {
      "id": "EV456",
      "created_at": "2014-08-03T12:00:00.000Z",
      "action": "failed",
      "resource_type": "payments",
      "links": {
        "payment": "PM456",
        "organisation": "OR456"
      },
      "details": {
        "origin": "bank",
        "cause": "mandate_cancelled",
        "description": "Customer cancelled the mandate at their bank branch.",
        "scheme": "bacs",
        "reason_code": "ARUDD-1"
      }
    }
  ]
}
```

## App Fees

If your OAuth app is creating payments on behalf of your users you may wish to add on a fee in addition to the fee that GoCardless charges.

When an OAuth app creates a payment or a subscription, it can specify an `app_fee`. This is an amount in pence or cents that will be deducted from the amount that the user receives. A fee charged by an app must be no more than 50% of the total payment amount.

App fees will be collated in a [payout](#) and paid out. App fee payouts are created and paid out daily, as with other payouts.



HTTP  
HTTP

```
POST https://api.gocardless.com/payments HTTP/1.1
Content-Type: application/json
{
  "payments": {
    "app_fee": 10,
    "amount": 100,
    "currency": "GBP",
    "links": {
      "mandate": "MD123"
    }
  }
}

POST https://api.gocardless.com/subscriptions HTTP/1.1
Content-Type: application/json
{
  "subscriptions": {
    "app_fee": 10,
    "amount": 2500,
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

```

    "currency": "GBP",
    "name": "Monthly Magazine",
    "interval_unit": "monthly",
    "day_of_month": "1",
    "links": {
      "mandate": "MA123"
    }
}
}

```

## Looking up an access token

Using the API you can query an access token to check if it is valid, and if so, to find out more about it (for example if you're not sure what scope the token has, or what organisation ID it relates to).

Relative endpoint: POST /oauth/introspect

### Parameters

`token`  
*required* The access token you want to look up.  
`client_id`  
*required* The `client_id` for your registered app.  
`client_secret`  
*required* The `client_secret` for your registered app.

### Response

Following the [OAuth token introspection spec](#), this endpoint responds with JSON as follows:

`active`  
A boolean representing whether the provided access token is a valid, active access token attached to your app  
`scope`  
The scope of the access token, as originally requested (either `read_write` or `read_only`). Only returned if the token is active.  
`token_type`  
The type of token you've been issued - this will be `bearer`. Only returned if the token is active.  
`organisation_id`  
The ID of the GoCardless account which this token gives you access to. You should store this for future use, as you will need it to identify to which of your users a [webhook](#) relates. Only returned if the token is active.  
`email`  
A contact email for the GoCardless account which this token gives you access to. Only returned if the token is active.

### Errors

In case of an error, the endpoint responds with JSON including an `error` and human-readable `error_description`. `error` may contain one of the following:

`invalid_request`  
You have not provided at least one of the required parameters (`client_id`, `client_secret` and `token`), or one of your parameters is of the wrong type.  
`invalid_client`  
The `client_id` and `client_secret` you provided to authenticate as your app were invalid or refer to a disabled app.



HTTP

HTTP

POST <https://connect.gocardless.com/oauth/introspect> HTTP/1.1

Content-Type: application/x-www-form-urlencoded

client\_id=jBo8XCUJLN01Mzya9vYS-7X5D&client\_secret=hzBBiZTnx8g1PLwvJoVIJa1&token=live\_y7VPT0dgFZtFaAS9V8HT3

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "active": true,
  "scope": "read_write",
  "token_type": "bearer",
  "email": "accounts@acme.com",
  "organisation_id": "OR123"
}
```

POST <https://connect.gocardless.com/oauth/introspect> HTTP/1.1

Content-Type: application/x-www-form-urlencoded

client\_id=jBo8XCUJLN01Mzya9vYS-7X5D&client\_secret=hzBBiZTnx8g1PLwvJoVIJa1&token=non\_existent\_or\_inactive\_token

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "active": false
}
```

## Disconnecting a user from your app

If a user no longer wishes to use your app, they can do this by disconnecting themselves using the API.

Relative endpoint: POST /oauth/revoke

### Parameters

`token`  
*required* The access token you want to revoke.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

validate) their access token

`client_id`  
*required* The `client_id` for your registered app.  
`client_secret`  
*required* The `client_secret` for your registered app.

## Response

Following the [OAuth token revocation spec](#), providing you successfully authenticate using your `client_id` and `client_secret`, this API will always respond with a `200 OK` and no body, **whether a token was successfully revoked or not**.

## Errors

In case of an error, the endpoint responds with JSON including an `error` and human-readable `error_description`. `error` may contain one of the following:

`invalid_request`  
 You have not provided at least one of the required parameters (`client_id`, `client_secret` and `token`), or one of your parameters is of the wrong type.  
`invalid_client`  
 The `client_id` and `client_secret` you provided to authenticate as your app were invalid or refer to a disabled app.



HTTP  
HTTP

```
POST https://connect.gocardless.com/oauth/revoke HTTP/1.1
Content-Type: application/x-www-form-urlencoded

client_id=jBo8XCUJLN01Mzya9vYS-7X5D&client_secret=hzBBiZTwnx8g1PLwvJoVIJa1&token=live_y7VPT0dgFZtFaAS9V8HT3

HTTP/1.1 200 OK
Content-Length: 0
```

## Compliance Requirements

When building a Direct Debit integration your payment pages and notifications need to comply with scheme-level rules. To help you design compliant pages quickly we have a series of guides:

### Payment pages:

- [ACH requirements \(standard\)](#)
- [ACH requirements \(advanced\)](#)
- [Autogiro requirements](#)
- [Bacs requirements](#)
- [BECS requirements](#)
- [BECS NZ requirements](#)
- [PAD requirements](#)
- [SEPA requirements](#)

### Customer notifications:

- [ACH requirements](#)
- [Autogiro requirements](#)
- [Bacs requirements](#)
- [BECS requirements](#)
- [BECS NZ requirements](#)
- [PAD requirements](#)
- [SEPA requirements](#)

Before your account can be set to “live” our sponsor bank will need to sign off your compliance in the above areas.

Alternatively, you may wish to use the [Redirect Flow API](#), which enables you to use payment pages hosted by GoCardless, and / or to have GoCardless send compliant, generic notification emails on your behalf.

You can see our hosted payment pages and an example of the emails we send by testing out the payment flow [here](#).

## Local Bank Details

[Creditor bank accounts](#), [customer bank accounts](#), and [customer bank account tokens](#) can all be created with local or international bank details. This section lists the different local bank detail formats, and how they should be used with the GoCardless API.

### IBAN

When supplying an IBAN (international bank account number), the `country_code` field is optional, but must match the country code of the IBAN (its first two characters) if supplied. If currency is not supplied we use the default currency based on the provided or derived `country_code`. For example if the IBAN of an account begins with GB but you w

**IBANs are not supported for Au**  
**details.**

HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "iban": "GB60 BARC 2000 00"
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

K - you must supply [local bank](#)



Consultez notre politique de confidentialité

## UK

For the UK, a 6-8 digit `account_number` and a 6 digit `branch_code` (sort code) should be supplied.

 HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "GB",
    "account_number": "55779911",
    "branch_code": "200000",
    ...
  }
}
```

## Australia

For Australia, an `account_number` and a `branch_code` (BSB number) should be supplied. The account number is 5-9 digits, and the BSB number is 6 digits.

**IBANs are not supported for Australian bank accounts denominated in AUD** - you must supply local bank details.

 HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "AU",
    "account_number": "0123456789",
    "branch_code": "012345",
    ...
  }
}
```

## Austria

For Austria, an `account_number` (Kontonummer, Kto.) and `bank_code` (Bankleitzahl, BLZ) should be supplied. The Kontonummer is 4-11 digits, and the Bankleitzahl is 5 digits.

 HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "AT",
    "account_number": "234573201",
    "bank_code": "19043",
    ...
  }
}
```

## Belgium

For Belgium, only an `account_number` (Rekeningnummer/Numéro de compte) should be supplied. The first three digits of this are a bank identifier, but are usually considered part of the account number.

 HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "BE",
    "account_number": "539-0075470-34",
    ...
  }
}
```

## Canada

For Canada, use a 2-3 digit `bank_code` (Financial Institution number), a 5 digit `branch_code` (Branch Transit number) and a 7-12 digit `account_number`.

 HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "CA",
    "account_number": "1234567",
    "bank_code": "0003",
    "branch_code": "00006",
    ...
  }
}
```

## Cyprus

For Cyprus, a 3 digit `bank_code` (Καταστήματος) can also be supplied.

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

5 digit `branch_code` (Kodikos

The bank and branch code (Kodikos Trapezas and Kodikos Katastimatos) are frequently supplied together as a single number, so it is also allowed to supply just an 8-digit `bank_code` containing both the Kodikos Trapezas and the Kodikos Katastimatos.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "CY",
    "account_number": "1200527600",
    "bank_code": "002",
    "branch_code": "00128",
    ...
  }
}
```

## Denmark

For Denmark, a 2-4 digit `bank_code` (registreringsnummer) and a 9-10 digit `account_number` (kontonummer) must be supplied.

Alternatively, the bank code and branch code can be provided as part of the `account_number` field separated by a - (e.g. 123-123-1234), or a 13-14-digit `account_number` (including bank code) can be provided where values are separated by a space (e.g. 1234 1234567890).



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "DK",
    "account_number": "3179681",
    "bank_code": "345",
    ...
  }
}
```

## Estonia

All local account numbers were replaced with IBANs in February 2014, however some payers may prefer using the old account numbers. These are up to 14 digits, and can be supplied as the `account_number`.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "EE",
    "account_number": "221020145685",
    ...
  }
}
```

## Finland

As of August 2014, Finnish banks are required to use IBANs rather than the old “tilinumeron rakenne”, however GoCardless still supports the older format. This is a 6 digit `bank_code`, and up to 8 digit `account_number`.

The Åland Islands use the same banking details as Finland, and can be submitted using their ISO 3166-1 country code AX.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "FI",
    "account_number": "456",
    "bank_code": "500015",
    ...
  }
}
```

## France

There are four parts to French bank details:

- Code banque: 5 digit `bank_code`
- Code guichet: 5 digit `branch_code`
- Numéro de compte: up to 14 digits
- Clé RIB: 2 check digits

The check digits are frequently written as part of the account number.

French DOMs and TOMs use the following codes:

- French Guiana: GF
- Guadeloupe: GP
- Martinique: MQ
- Mayotte: YT
- Réunion: RE
- Saint Barthélemy: BL
- Saint Martin (French part): MF

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

ber field.

- Saint Pierre and Miquelon: PM



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "FR",
    "account_number": "0500013M026 06",
    "bank_code": "20041",
    "branch_code": "01005",
    ...
  }
}
```

## Germany

For Germany, a 1-10 digit `account_number` (Kontonummer) and 8 digit `bank_code` (Bankleitzahl) are required.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "DE",
    "account_number": "532013000",
    "bank_code": "37040044",
    ...
  }
}
```

## Greece

In Greece most customers know their IBANs, but GoCardless also supports passing full local bank details. If local bank details are provided, a 3 digit `bank_code` (Kodikos Trapezas), a 5 digit `branch_code` (Kodikos Katastimatos), and a 16 digit `account_number` (Arithmos Logiasmou) must be supplied.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "GR",
    "account_number": "0000000012300695",
    "bank_code": "011",
    "branch_code": "0125",
    ...
  }
}
```

## Ireland

Ireland uses a 6-digit `branch_code` (sort code), and 6-8 digit `account_number`.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "IE",
    "account_number": "29823529",
    "branch_code": "90-21-27",
    ...
  }
}
```

## Italy

Most Italian payers will be familiar with the IBAN format in addition to the local format, and more correctness checks can be applied to the IBAN, so it is recommended to ask users to input their IBAN rather than local details. However, it is possible to use:

- Codice ABI: 5 digit `bank_code`
- CAB: 5 digit `branch_code`
- Numero di conto: up to 12 digit `account_number`



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "IT",
    "account_number": "123456",
    "bank_code": "05428",
    "branch_code": "11101",
    ...
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

[Consultez notre politique de confidentialité](#)

## Latvia

Latvia replaced bank details with IBANs in 2005, so most Latvian payers are unlikely to know their bank code and account number, however GoCardless continues to accept them. The `bank_code` should be 4 characters, and the `account_number` should be 13 digits.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "LV",
    "account_number": "0000435195001",
    "bank_code": "LACB",
    ...
  }
}
```

## Lithuania

Lithuania replaced bank details with IBANs in 2004, so most Lithuanian payers are unlikely to know their bank code and account number, however GoCardless continues to accept them. The `bank_code` should be 5 digits, and the `account_number` should be 11 digits.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "LT",
    "account_number": "11101001000",
    "bank_code": "10000",
    ...
  }
}
```

## Luxembourg

Luxembourg replaced bank details with IBANs in 2002, so most Luxembourgian payers are unlikely to know their bank code and account number, however GoCardless continues to accept them. The `bank_code` should be 3 digits, and the `account_number` should be 13 digits.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "LU",
    "account_number": "9400644750000",
    "bank_code": "001",
    ...
  }
}
```

## Malta

Malta uses a 5-digit `branch_code` (sort code) and an up to 18-digit `account_number` (numru tal-kont bankarju).



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "MT",
    "account_number": "9027293051",
    "branch_code": "44093",
    ...
  }
}
```

## Monaco

Monaco uses the same local bank details as France. That is:

- Code banque: 5 digit `bank_code`
- Code guichet: 5 digit `branch_code`
- Numéro de compte: up to 11 digit `account_number`
- Clé RIB: 2 check digits

The check digits are frequently written at the end of the `account_number`, and this is how they should be included in the `account_number` field.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "MC",
    "account_number": "05000013",
    "bank_code": "20041",
    "branch_code": "01005",
    ...
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

## Netherlands

The Netherlands have replaced local bank details with IBANs, so most Dutch payers are unlikely to know their bank code and Rekeningnummer, however GoCardless continues to accept them. The `bank_code` should be 4 characters, and the `account_number` (Rekeningnummer) up to 10 digits.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "NL",
    "account_number": "0417164300",
    "bank_code": "ABNA",
    ...
  }
}
```

## New Zealand

New Zealand bank details have 4 components:

- Bank Number: 2 digit `bank_code`
- Branch Number: 4 digit `branch_code`
- Account Number: 7 digits
- Account Suffix: 2/3 digits

The account number and account suffix should be concatenated with a - for the `account_number` field.

**IBANs are not supported for New Zealand bank accounts denominated in NZD** - you must supply local bank details.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "NZ",
    "account_number": "0123456-89",
    "branch_code": "0123",
    "bank_code": "01",
    ...
  }
}
```

## Portugal

Portuguese bank details have 4 components:

- Código de Banco: 4 digit `bank_code`
- Código de Balcão: 4 digit `branch_code`
- Número de conta: 11 digit account number
- Dígitos de controlo: 2 check digits

The account number and check digits should be concatenated for the `account_number` field.



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "PT",
    "account_number": "1234567890154 50",
    "bank_code": "0002",
    "branch_code": "0123",
    ...
  }
}
```

## San Marino

San Marino uses the same bank details formats as Italy. As with Italy, payers will be familiar with the IBAN format in addition to the local format, and more correctness checks can be applied to the IBAN, so it is recommended to ask users to input their IBAN rather than local details. However, it is possible to use:

- Codice ABI: 5 digit `bank_code`
- CAB: 5 digit `branch_code`
- Numero di conto: up to 12 digit `account_number`



HTTP

HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "SM",
    "account_number": "123456",
    "bank_code": "05428",
    "branch_code": "11101",
    ...
  }
}
```

## Slovak Republic

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

There are three parts to Slovak bank details:

- Předčíslí: up to 6 digit account number prefix
- Číslo účtu: up to 8 digit account number
- Kód banky: 4 digit bank code

These are often written as pppppp-aaaaaaaa/bbbb. When creating a bank account using Slovak bank details, the předčíslí and číslo účtu should be concatenated for the account\_number field, and the kód banky should be used for the bank\_code field.



HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "SK",
    "account_number": "198742637541",
    "bank_code": "1200",
    ...
  }
}
```

## Slovenia

Slovenia uses a 5 digit bank\_code and up to 10 digit account\_number.



HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "SI",
    "account_number": "123438",
    "bank_code": "19100",
    ...
  }
}
```

## Spain

There are four parts to Spanish bank details, the 4 digit código de entidad (bank code), 4 digit código de oficina (branch code), 2 dígitos de control (check digits), and 10 digit número de cuenta.

These can either be supplied as separate bank\_code, branch\_code, and account\_number (with the check digits at the start of the account number), or all in the account\_number field. In the latter case, this is known locally as the Código Cuenta Cliente.



HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "ES",
    "account_number": "45 0200051332",
    "bank_code": "2100",
    "branch_code": "0418",
    ...
  }
}
```

## Sweden

Sweden uses a 4- or 5-digit branch\_code (clearingnummer), and up to 10-digit account\_number (kontonummer).

**IBANs are not supported for Swedish bank accounts denominated in SEK** - you must supply local bank details.



HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "SE",
    "account_number": "1234512",
    "branch_code": "5527",
    ...
  }
}
```

## United States

United States uses a 9-digit bank\_checking or savings).

**IBANs are not supported for Ar**



HTTP  
HTTP

```
{
  "customer_bank_accounts": {
    "country_code": "US",
    "account_number": "2715500356",
    ...
  }
}
```

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.

\_type must be provided (either

Consultez notre politique de confidentialité

```

"bank_code": "026073150",
"account_type": "checking",
...
}

```

## Character Sets

Different schemes allow different sets of characters to be used in [payment](#) and [mandate](#) references. All schemes allow alphanumeric characters (A-Z, 0-9) and some also allow a selection of special characters (subject to change). Any references provided to us containing characters not permitted by the scheme will cause an error, we will include the characters that caused the error in the message.

The exact way that references are displayed also varies by bank (e.g. some truncate references, some do not show them at all). GoCardless will always format payment and mandate references to appear as accurately as possible for the given bank and scheme.

## Public Certificate Policy

GoCardless forces HTTPS for all services, using TLS. Our client libraries also use TLS and verify the certificates to ensure security.

Currently our certificates are provided by Google (Google GTS Root R1 CA). This can be verified using the `openssl` library:

```

$ openssl s_client -connect api.gocardless.com:443 <<< ""
CONNECTED(00000005)
depth=2 C = US, O = Google Trust Services LLC, CN = GTS Root R1
verify return:1
depth=1 C = US, O = Google Trust Services LLC, CN = GTS CA 1P5
verify return:1
depth=0 CN = api.gocardless.com
verify return:1
---
Certificate chain
0 s:CN = api.gocardless.com
    i:C = US, O = Google Trust Services LLC, CN = GTS CA 1P5
        a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
        v:NotBefore: Jun 15 13:55:41 2023 GMT; NotAfter: Sep 13 13:55:40 2023 GMT
1 s:C = US, O = Google Trust Services LLC, CN = GTS CA 1P5
    i:C = US, O = Google Trust Services LLC, CN = GTS Root R1
        a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
        v:NotBefore: Aug 13 00:00:42 2020 GMT; NotAfter: Sep 30 00:00:42 2027 GMT
2 s:C = US, O = Google Trust Services LLC, CN = GTS Root R1
    i:C = BE, O = GlobalSign nv-sa, OU = Root CA, CN = GlobalSign Root CA
        a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA256
        v:NotBefore: Jun 19 00:00:42 2020 GMT; NotAfter: Jan 28 00:00:42 2028 GMT
---

```

Our [client libraries](#) currently don't offer any form of certificate pinning, and we don't intend to support this feature in the future. **We reserve the right to change our Certificate Authority (CA), which will break your integration if you've implemented certificate pinning..**

For any future changes to the API certificate authority, we will place a scheduled maintenance notice on our [status page](#), with at least 2 weeks advance warning. If your integration does depend on a specific TLS chain of trust, then we advise that you subscribe to updates on this page.

## Approving our IP Addresses

Please be aware that adding our IPv4 addresses to an approved list is not a supported scenario for using our API. The IP address may change as part of our normal operations and should not be relied upon to remain static. This is inline with best practices for highly-available services.

As a result we recommend that you do not add our IPv4 addresses to your approved lists in the future.

Approving our IP addresses for receiving webhooks is still a valid practice. Please visit the [webhooks appendix](#) for more details.

## Alternative to IP approved lists

We recommend relying on the stronger security provided by Transport Layer Security (TLS). This should be used to validate our identity whilst communicating with our API as part of the normal checking of certificates. Please note that certificate revocation should also be checked for all parts of the certificate chain in the certificate we provide you.

For more information on how to implement this change, or for help with any questions, please contact our team on [api@gocardless.com](mailto:api@gocardless.com).

## Tax Rates

GoCardless has been instructed by the relevant authorities to charge tax on its fees.

GoCardless currently applies tax to transaction and surcharge fees for merchants operating in the UK and France.

### Who does this impact?

At this time, this change will impact other areas that we operate in the

In the onboarding flow we request

### Best practices for integratin

For best practices see our [partner](#)

### Definitions of jurisdiction

jurisdiction for countries or overs

#### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité

t charging VAT (or equivalent) in

rect tax to the merchant.

For areas with unique tax rates that do not have ISO-3166-1 codes, we will use the [ISO 3166-2 subdivision codes](#) for that country. e.g Corsica is FR-COR.

## Tax Tables

### Tax for British territories

<b>id</b>	<b>jurisdiction type</b>	<b>description</b>	<b>percentage</b>
GB_VAT_1	GB	VAT VAT for Mainland Great Britain	20
JE_GST_1	JE	GST GST for Jersey	0
GL_VAT_1	GI	VAT VAT for Gibraltar	0
GG_VAT_1	GG	VAT VAT for Guernsey	0

### Tax for French territories

<b>id</b>	<b>jurisdiction type</b>	<b>description</b>	<b>percentage</b>
FR_VAT_1	FR	VAT VAT for Mainland France	20
GP_VAT_1	GP	VAT VAT for Guadeloupe	8.5
MQ_VAT_1	MQ	VAT VAT for Martinique	8.5
RE_VAT_1	RE	VAT VAT for Reunion Island	8.5
FR-COR_VAT_1	FR-COR	VAT VAT for Corsica	20

The ids for tax rates are shared between the environments.

## Security Requirements

### Accessing Bank Details

There are certain security requirements which apply to GoCardless when providing access to bank account details, and to you (the user) when using upgrades which grant access to bank account details. Currently the following upgrades grant access to bank account details:

- Encrypted Bank Details Access
- Transfer Bank Accounts

#### GoCardless' responsibilities:

- Implement procedures and technical controls that ensure secure storage and transmission of personal data using industry-standard encryption mechanisms and strong cipher suites
- Implement technical controls to ensure that any data sent to, and received from, intended recipients is protected against eavesdropping, modification and loss in transit
- Apply access controls to ensure that only Authorised Users gain access to information systems that process and store key material
- Maintain and monitor audit logs on the use of the Transfer Bank accounts and Encrypted Bank Details Access features
- Have policies and procedures to detect and respond to security incidents, including procedures to monitor systems, mitigate effects, and document incidents and their outcomes
- Maintain Developer Documentation concerning the use, creation, management and revocation of encryption keys, access tokens and digital certificates
- Keep a backup of encryption key material and related business-critical data and have appropriate recovery and continuity procedures

#### Your responsibilities:

- Implement and maintain procedures and technical controls to ensure that only Authorised Users access systems that are used to manage and store key material that is shared with GoCardless
- Manage key material by implementing the creation, storage, rotation, revocation and disposal requirements set in GoCardless' Developer Documentation
- Generate RSA keys at least 2048-bit in length with SHA256
- Rotate keys at least annually
- Use industry-standard encryption mechanisms to transmit and store encryption data used in the Transfer Bank Accounts and Encrypted Bank Details Access features and GoCardless' services
- Establish and maintain procedures to back up your key material, detect when backup failures occur, and take corrective action for recovery as required

### Nous utilisons des cookies

Les cookies nous permettent d'optimiser le fonctionnement de notre site, d'améliorer nos produits ainsi que de vous offrir une meilleure expérience de navigation. Vous pouvez gérer les options si vous souhaitez modifier la sélection, ou vous pouvez également accepter tous les cookies, y compris les cookies de publicité.



Consultez notre politique de confidentialité