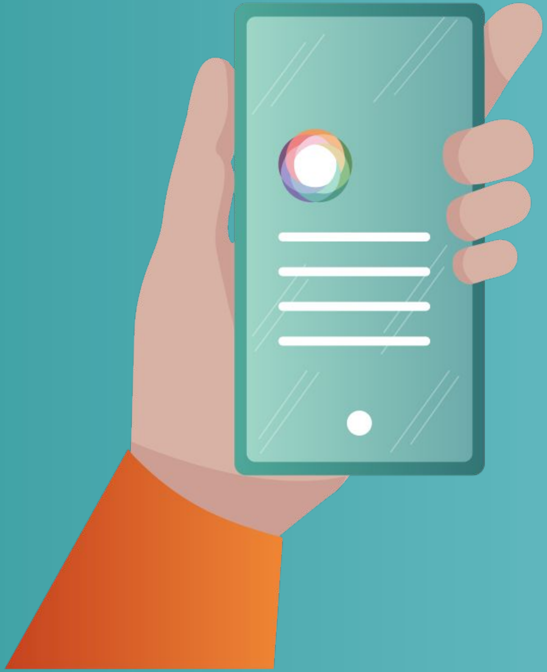
Interledger

Open Payments
Third party account access

Sarah Jones

TypeScript Developer, ILF

10-06-2024

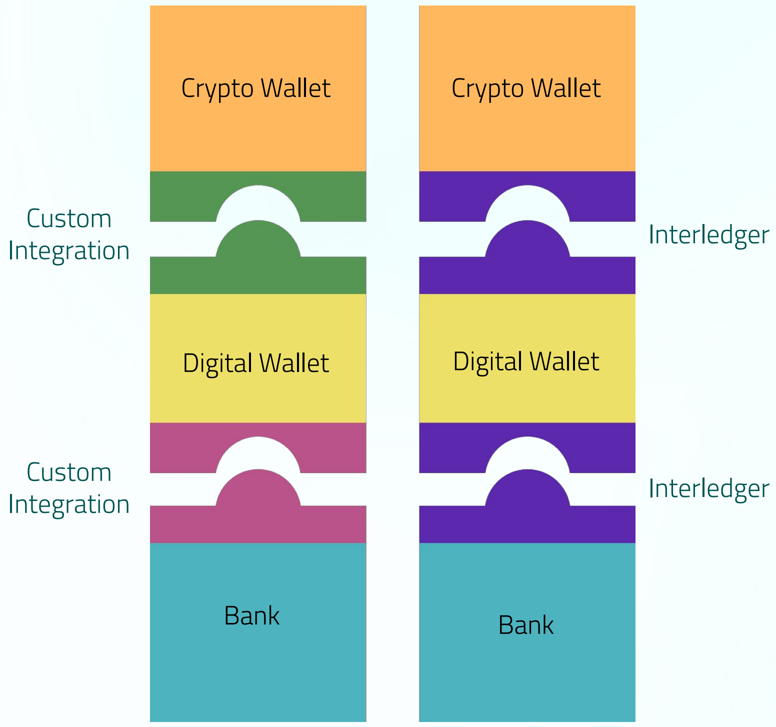


The Status Quo

- ➔ Financial institutions have their own internal systems, making direct money transfers between different systems difficult.
- ➔ Either connect via:
  - ◆ custom integrations, which are costly, time-consuming, and prone to errors.
  - ◆ intermediaries, which increases costs and adds delays.

Where Interledger Comes In

Standardisation



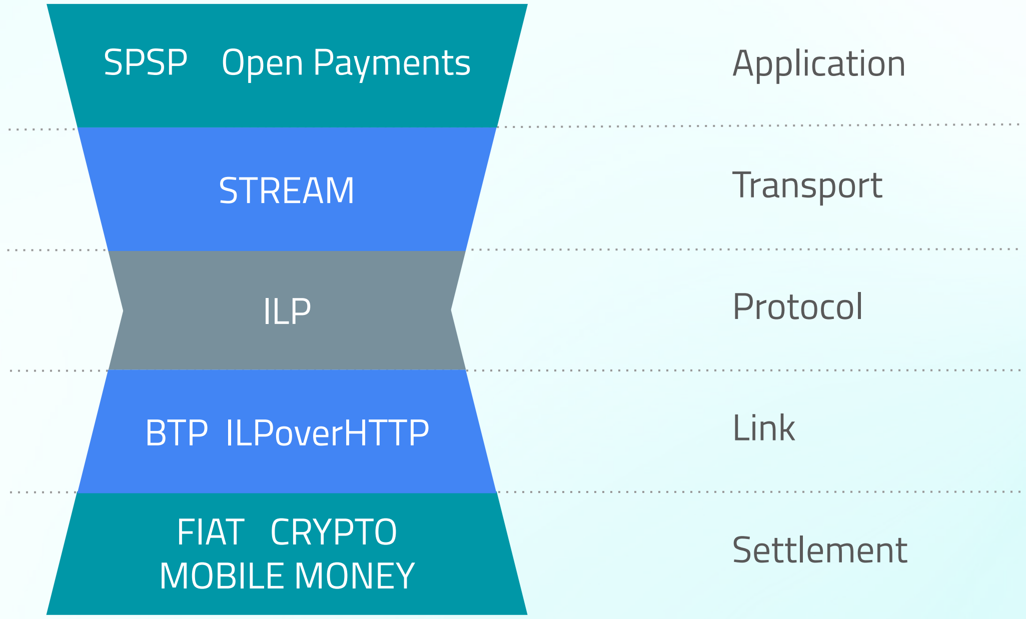
→ Open protocol for seamless value transfer.

→ It gives us scalable, secure, interoperability.

Intro to the Interledger Protocol (ILP)

The Protocol Suite

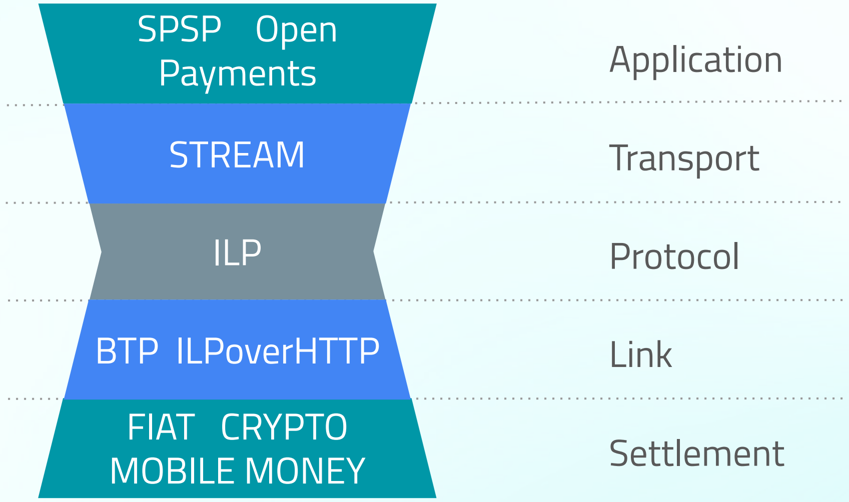
- Protocol suite
- Layered architecture

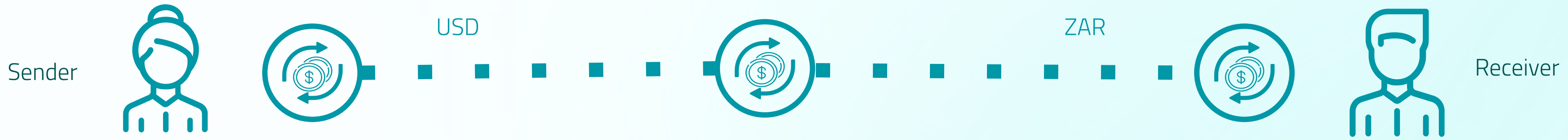


Intro to the Interledger Protocol (ILP)

The Protocol Suite

- ➔ Ledgers: outside of ILP suite. Are the systems that keep track of value.
- ➔ Connectors: route packets of value across different ledgers.
- ➔ ILP Packets: payments are broken down into smaller packets.





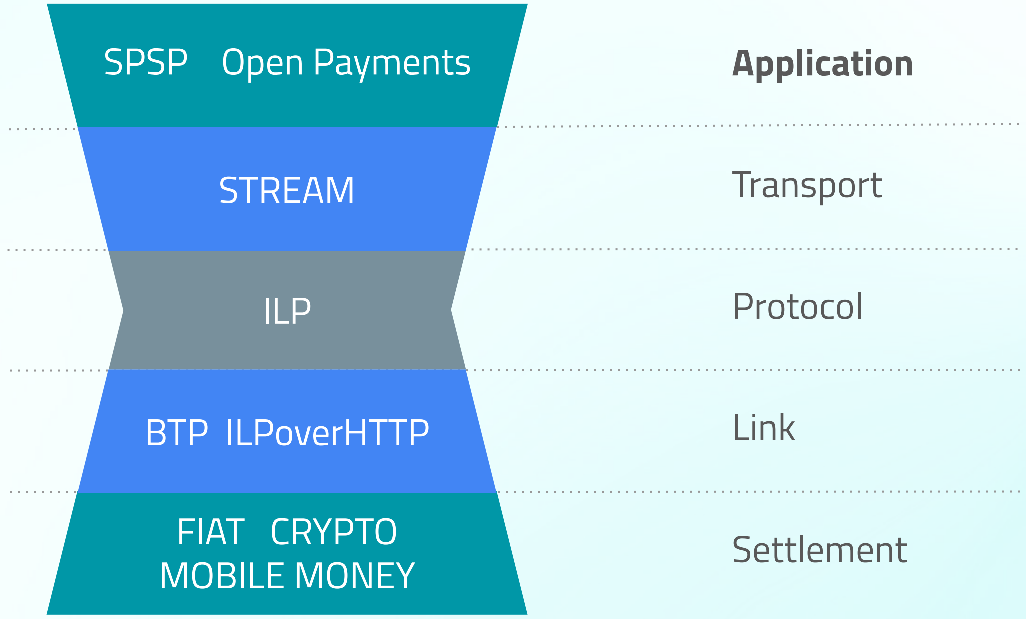
How Interledger Adds Value

- ➔ Enabling:
  - ◆ Cross-Border payments
  - ◆ Cross-currency payments
  - ◆ Financial inclusion - access to digital finance or better digital finance
  - ◆ Micropayments - sending small payments (fractions of a cent)

Intro to the Interledger Protocol (ILP)

The Application Layer

- ➔ This is a standard for user-facing applications
- ➔ Destination account discovery
- ➔ Destination amount negotiation
- ➔ Transport protocol selection and communication of associated details.



Intro to the Interledger Protocol (ILP)

Takeaways

- ➔ Financial systems are isolated, needing intermediaries or costly custom integrations for transfers.
- ➔ Standards make interoperability easier.
- ➔ Interoperability has a lot of benefits, from digital financial inclusion to revolutionising digital financial experiences like streaming micropayments.
- ➔ Utilizes existing payment rails.
- ➔ Layered architecture approach.
- ➔ Means we can use Open Payments on top of all of this without knowing how it all works anyway!

Open Payments (OP)

What It Is & Why It Matters

- ➔ Enables applications to communicate directly with their customer’s accounts.
- ➔ Sends payment instructions, not money (no need to be a financial service provider)
- ➔ Distributed, federated, global payment ecosystem
- ➔ No dependency on specific account types or settlement systems
- ➔ Developers can build payment functionality:
  - ◆ Without third-party payment processors
  - ◆ Without custom integrations
  - ◆ In any programming language

What is Open Payments (OP)

Terminology

- ➔ Account Servicing Entity (ASE)
  - ◆ Bank, mobile money provider, digital wallet provider etc.
  - ◆ Are regulated entities within the countries they operate.
  - ◆ They manage their customer's financial accounts

What is Open Payments (OP)

Terminology

- API: Application Programming Interface
- Defines how different services communicate with each other
- How It Works:
  - Request: One application sends a request for data or action.
  - Response: The other application sends back the requested data or performs the action.
- E.g. A surf website (client) needs to make a call to a weather service (server) to get information to use to suggest the best days and locations for surfing.

What is Open Payments (OP)

Terminology

- ➔ API standard -> words on a page
- ➔ E.g. retrieve a payment:

Name	Type	Explanation
walletAddress required	string	URL of a wallet address they money is being sent to
quoteId required	string	The URL of the quote defining this payment's amount
metadata	object	Additional metadata associated with the payment

What is Open Payments (OP)

Terminology

- RESTful API (Representational State Transfer) -> code
- Stateless: Each request from a client to server must contain all the information needed to understand and process the request.
- RESTful APIs use standard HTTP methods to perform operations on resources
  - GET: Retrieve a resource.
  - POST: Create a new resource.
  - PUT: Update an existing resource.
  - DELETE: Remove a resource.
  - PATCH: Partially update a resource.

What is Open Payments (OP)

- ➔ OP enables direct payment integration between applications and user’s accounts
- ➔ It is an API standard
- ➔ RESTful API
- ➔ It requires:
  - ◆ Account servicing entities (ASEs) to create Open Payments-enabled accounts
  - ◆ A common payment rail between the ASEs

Who uses Open Payments (OP)

Interledger FOUNDATION

Third party applications call the OP APIs

- ➔ Use OP to interact with customer accounts.
- ➔ Connect to any ASE using OP without custom integrations.
- ➔ E.g. send a request to make a quote for a customer who wants to download a song

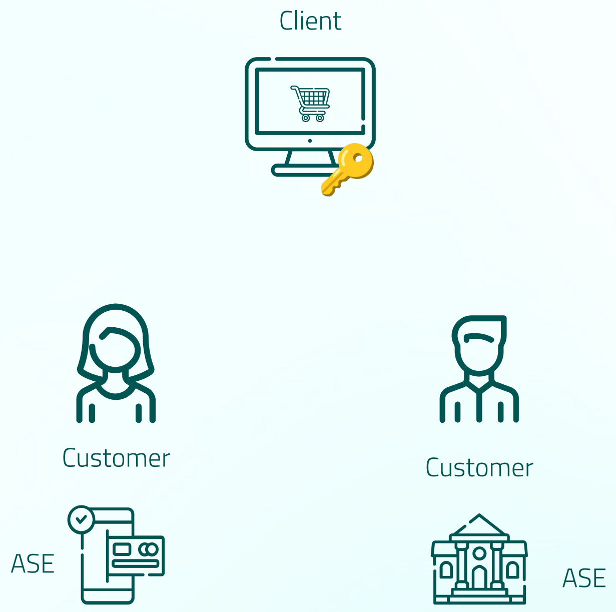
ASEs implement the OP standard

- ➔ Write the code to implement the functionality behind the API requests.
- ➔ E.g. When a request is received to make a quote, gather all necessary information, calculate any fees, determine validity period, add a new entry to the database, etc.

Open Payments

Third-Party Access to Accounts

- → Grant Request Incoming Payment
- → Create Incoming Payment
- → Grant Request Quote
- → Create Quote
- → Grant Request Outgoing Payment
- → Continuation Request
- → Create Outgoing Payment

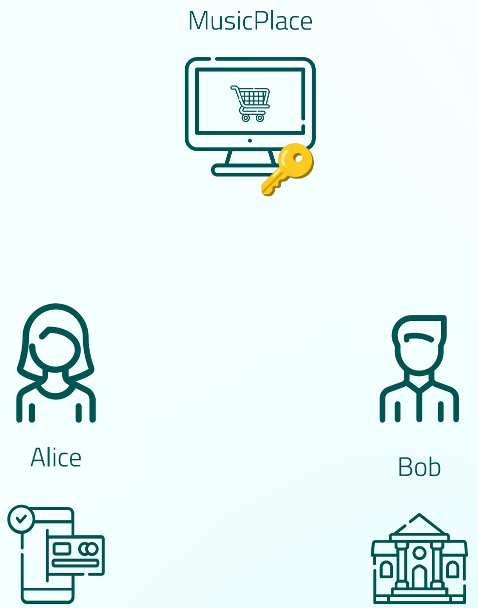


- Customer: end-users
- Client: app that processes payments by making OP API requests
- ASE: entity that manages customers' accounts and implements the OP standard.

Open Payments

Third-Party Access to Accounts

- → Grant Request Incoming Payment
- → Create Incoming Payment
- → Grant Request Quote
- → Create Quote
- → Grant Request Outgoing Payment
- → Continuation Request
- → Create Outgoing Payment



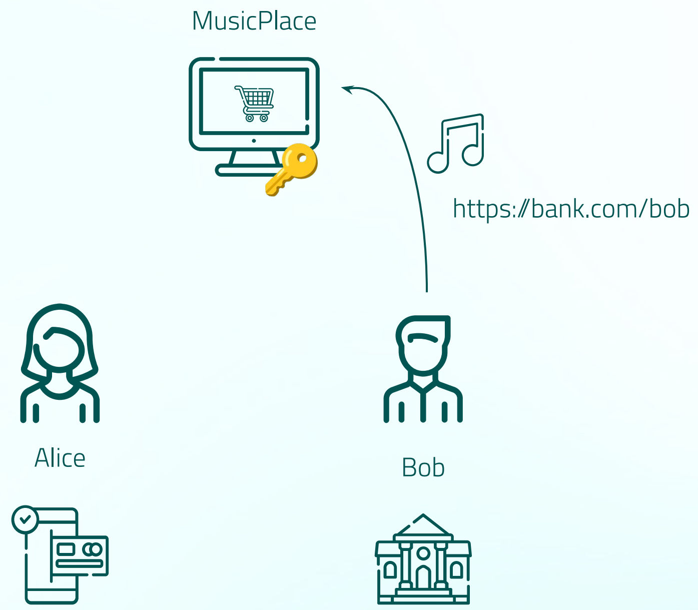
Who's who:

- Customers are Alice and Bob (Marley)
- Client is MusicPlace
- ASEs are a digital wallet and a bank

Open Payments

Third-Party Access to Accounts

- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment



Three participants:

- Bob add song to MusicPlace
- Provides his wallet address to be paid when someone downloads the song

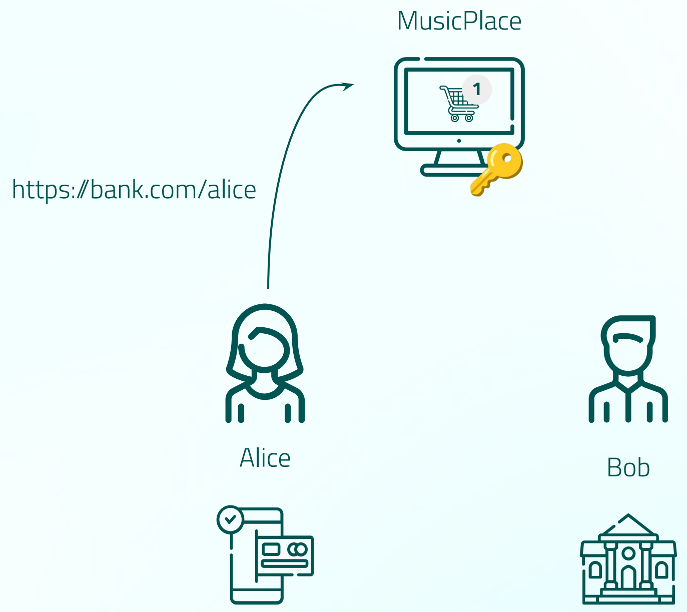
Interlude: Wallet Address

- ➔ Every OP-enabled account is identified by one or more URLs called wallet addresses
- ➔ A wallet address is an alias for the underlying financial account.
- ➔ A wallet address is a human readable, publicly shareable account identifier
- ➔ It is also a service endpoint that provide the entry point for the API.
- ➔ The payer and payee must both have a wallet address to transact using Open Payments.

Open Payments

Third-Party Access to Accounts

- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment

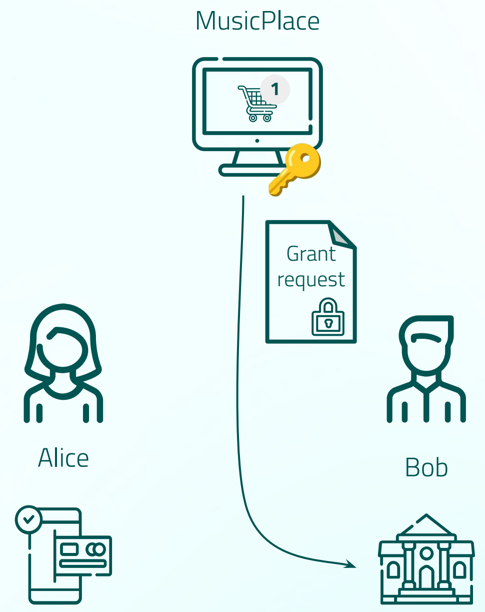


- Alice adds Bob’s song to her cart
- Provides her wallet address to pay for the song using Open Payments

Open Payments

Third-Party Access to Accounts

- ➔ **Grant Request Incoming Payment**
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment



- MusicPlace requests a **Grant** to create an **Incoming Payment** on **Bob's account** at his bank

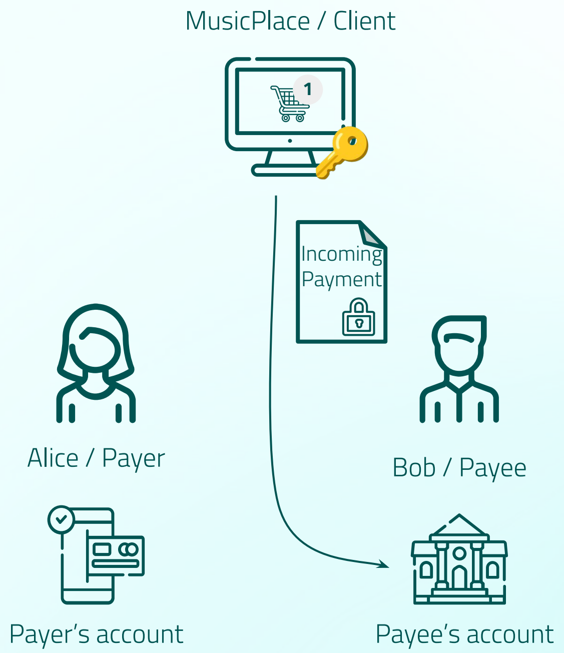
Interlude: Grants

Grant **N**egotiation and **A**uthorization **P**rotocol

- ➔ Clients must receive grants before issuing payment instructions.
- ➔ Grants give clients the authorization, via access tokens, to perform operations.
- ➔ All requests require signatures, which protect the integrity of the requests. Signatures are generated according to the HTTP Signatures specification.
- ➔ GNAP makes it possible to give account holders specific and fine-grained control over the permissions they grant, including control over the amounts of transactions with time-based and velocity-based limits. This enables third-party payment initiation and delegated authorization without compromising the security of the underlying financial accounts and payment instruments.

Interlude: Incoming Payment

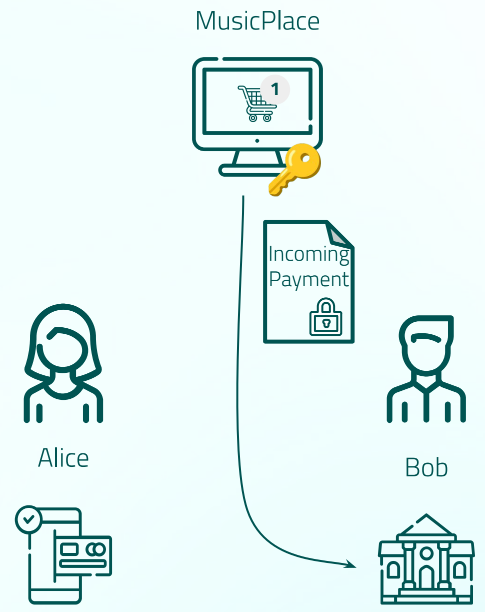
- ➔ The process begins by creating an incoming-payment on the payee's account.
- ➔ When created, the payee's ASE returns unique payment details to the client that can be used to address payments to the account.



Open Payments

Third-Party Access to Accounts

- ➔ Grant Request Incoming Payment
- ➔ **Create Incoming Payment**
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment

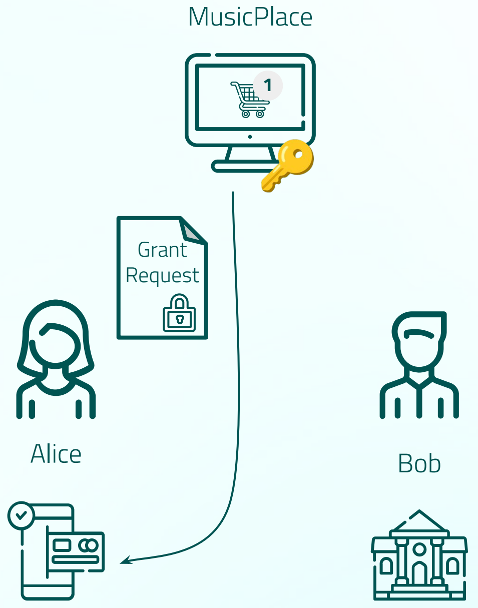


- MusicPlace creates an **Incoming Payment** on **Bob’s account** at his bank

Open Payments

Third-Party Access to Accounts

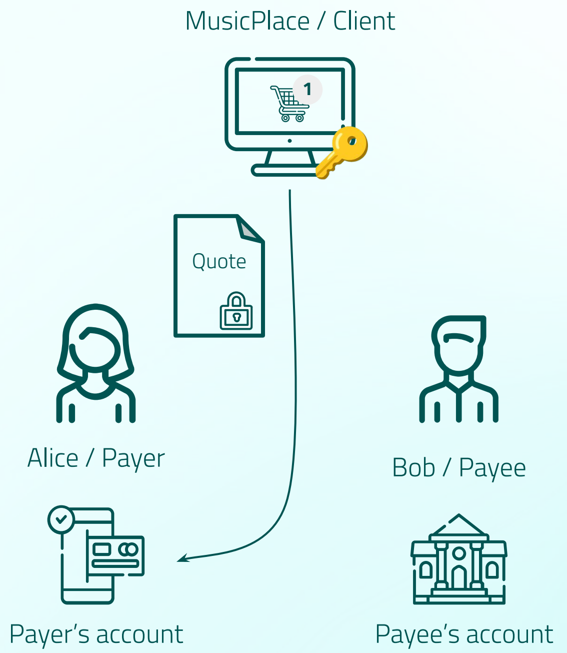
- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ **Grant Request Quote**
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment



- MusicPlace requests a **Grant** to create a **Quote** for the payment on **Alice's account** at her wallet provider

Interlude: Quote

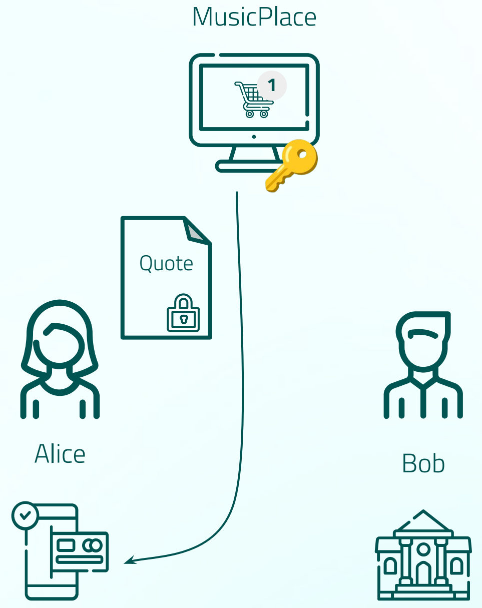
- ➔ After an incoming-payment is created on the payee's account, a quote must be created on the payer's account.
- ➔ The purpose of a quote is to indicate how much the transaction will cost, including fees.
- ➔ The quote is a commitment from the payer's ASE to deliver that amount to the payee's ASE.
- ➔ A quote is only valid for a limited time.



Open Payments

Third-Party Access to Accounts

- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ **Create Quote**
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment

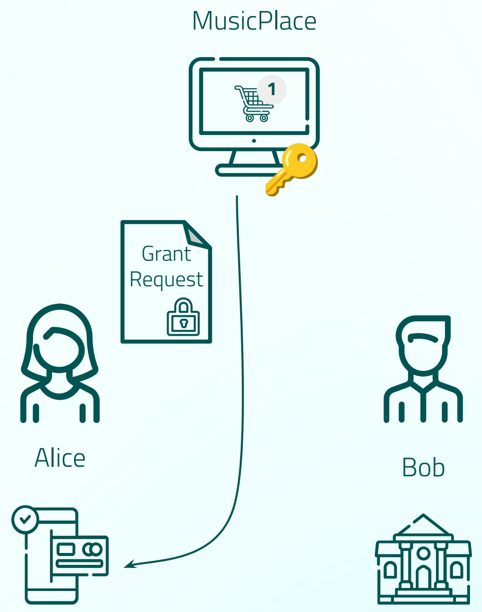


- MusicPlace creates a **Quote** for the payment on **Alice's account** at her wallet provider

Open Payments

Third-Party Access to Accounts

- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ **Grant Request Outgoing Payment**
- ➔ Continuation Request
- ➔ Create Outgoing Payment



- MusicPlace requests a **Grant** to create an **Outgoing Payment** on **Alice's account** at her wallet provider

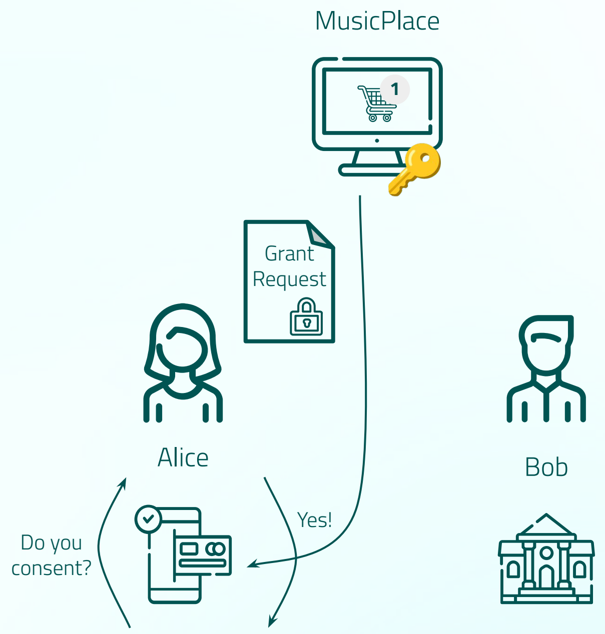
Interlude: Outgoing Payment

- After a quote is created Open Payments requires the payer to explicitly consent to the creation of an outgoing-payment.
- Consent is obtained through an interactive grant.
- The outgoing-payment serves as an instruction to make a payment from the payer’s account.
- The request to create the outgoing-payment includes:
  - the payee’s wallet address, so the payer’s ASE knows where to send the payment
  - the quote ID, where the payment amounts are defined.

Open Payments

Third-Party Access to Accounts

- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ **Continuation Request**
- ➔ Create Outgoing Payment

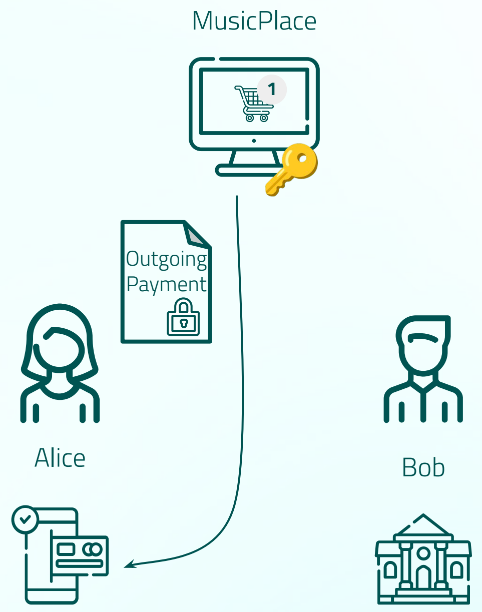


- Alice's wallet provider asks her to **approve** the payment

Open Payments

Third-Party Access to Accounts

- → Grant Request Incoming Payment
- → Create Incoming Payment
- → Grant Request Quote
- → Create Quote
- → Grant Request Outgoing Payment
- → Continuation Request
- → **Create Outgoing Payment**

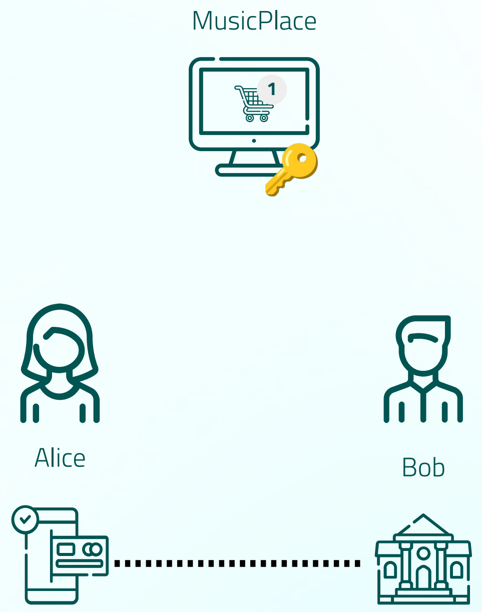


- MusicPlace creates an **Outgoing Payment** on **Alice's account** at her wallet provider

Open Payments

Third-Party Access to Accounts

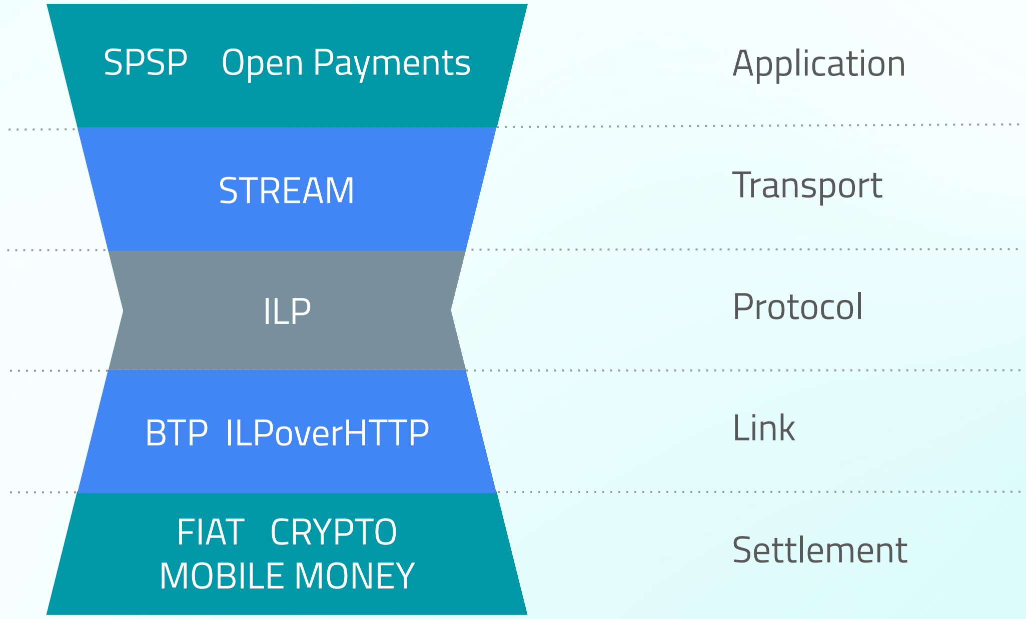
- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment



- **Payment** is sent **from Alice's account** at her wallet provider **to Bob's account** at his bank

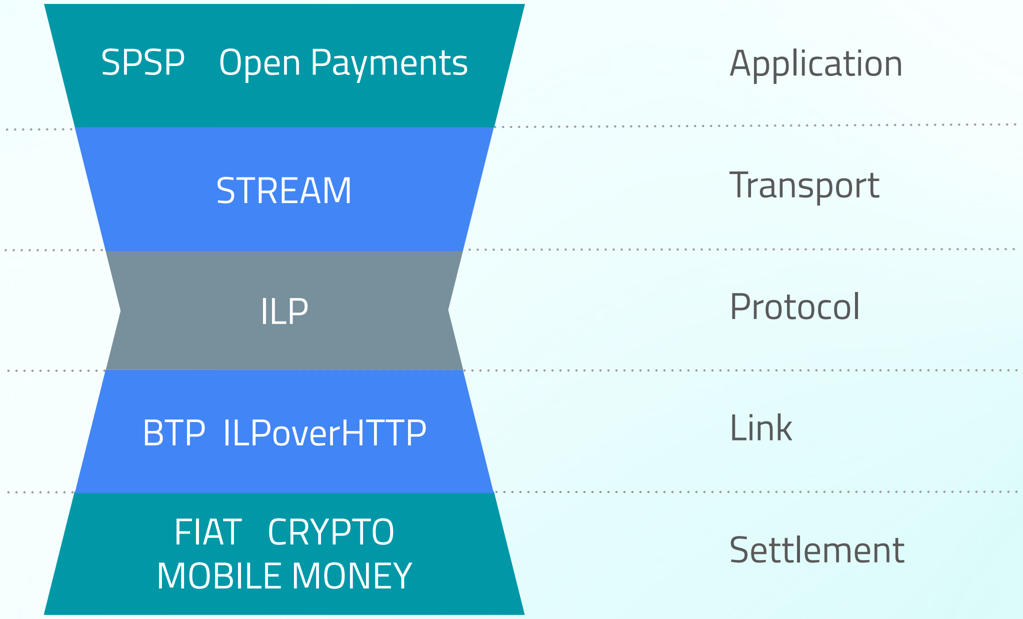
Zooming in on the dotted line

➔ The depiction of money moving between payer and payee is happening below the Open Payments layer.



How Open Payments leverages ILP

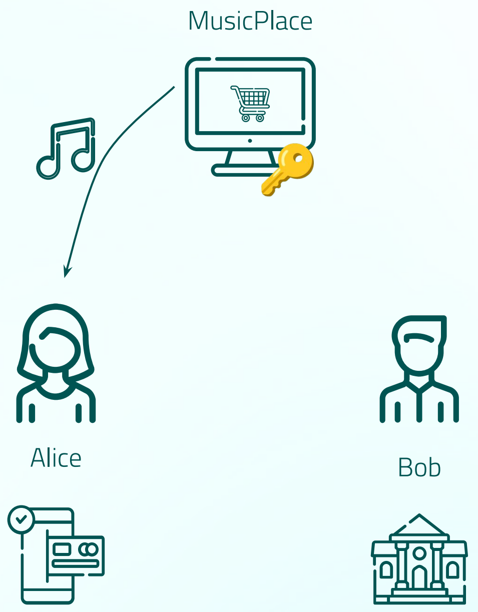
- ➔ OP results in a payment instruction.
- ➔ We still need a way to get that instruction back to the settlement entities.
- ➔ ILP protocol suite can take it, break those payment instructions up into small packets and send them over the network.



Open Payments

Third-Party Access to Accounts

- ➔ Grant Request Incoming Payment
- ➔ Create Incoming Payment
- ➔ Grant Request Quote
- ➔ Create Quote
- ➔ Grant Request Outgoing Payment
- ➔ Continuation Request
- ➔ Create Outgoing Payment



- Alice can download Bob’s music

How would this look without OP?

From The Client Application's Perspective

- ➔ If someone gives you their bank account / credit card info you can't do anything with it, even if they ask you to. You're not authorized and you don't have access.
- ➔ You can use a payment processor (e.g. Stripe or PayPal)
  - ◆ Adds a middleman with fees and delays
- ➔ Otherwise become a payment processor and integrate directly with banks
  - ◆ Technical complexity
  - ◆ Compliance complexity
- ➔ Handling sensitive financial data directly involves significant security risks.

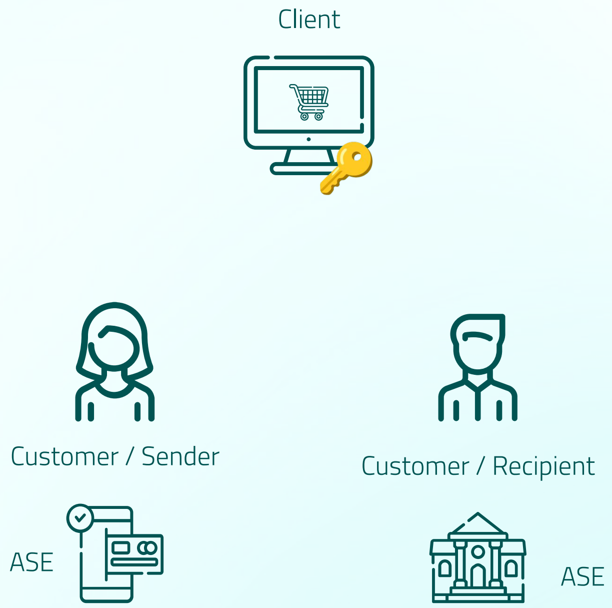
Advantages of using OP

- ➔ Users can grant specific access to their accounts for certain amounts and actions.
- ➔ Not limited to specific payment methods or providers.
- ➔ Standardisation makes integration lives easier. Can lead to cheaper transaction costs (remove middlemen, separates clearance from settlement, introduces competition into the ecosystem)
- ➔ By separating payment instructions from execution/settlement, app developers can include payment functionality without registering as a licensed financial services provider.
- ➔ Apps don't need to store sensitive info.

Deep Dive: Open Payment Flows

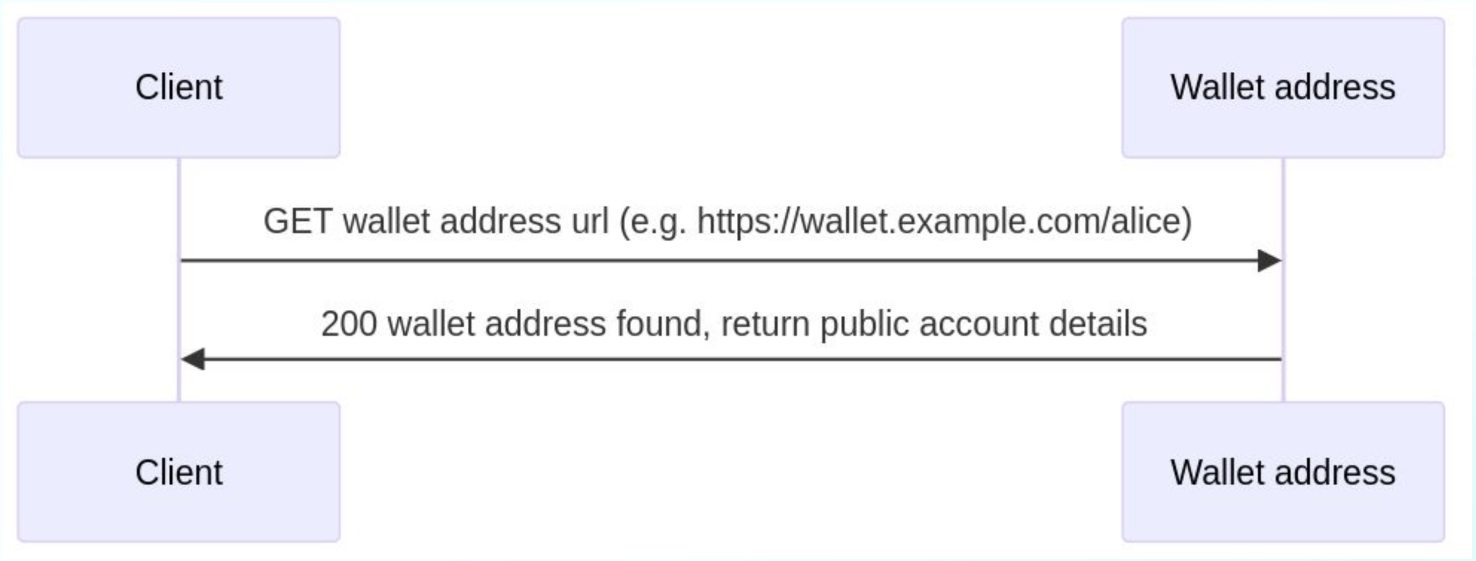
Who’s Who

- ➔ Client
- ➔ Customer
- ➔ ASE
  - ◆ Wallet address server
  - ◆ Resource server (RS)
  - ◆ Authorization server (AS)
  - ◆ Identity provider (IdP)



Open Payment Flows

Wallet address


Client
Wallet address
GET wallet address url (e.g. https://wallet.example.com/alice)
200 wallet address found, return public account details
Client
Wallet address

➔ Using URLs as payment instruments solves two of the biggest issues with existing payments UX: discoverability and interaction.

Open Payment Flows

Wallet address

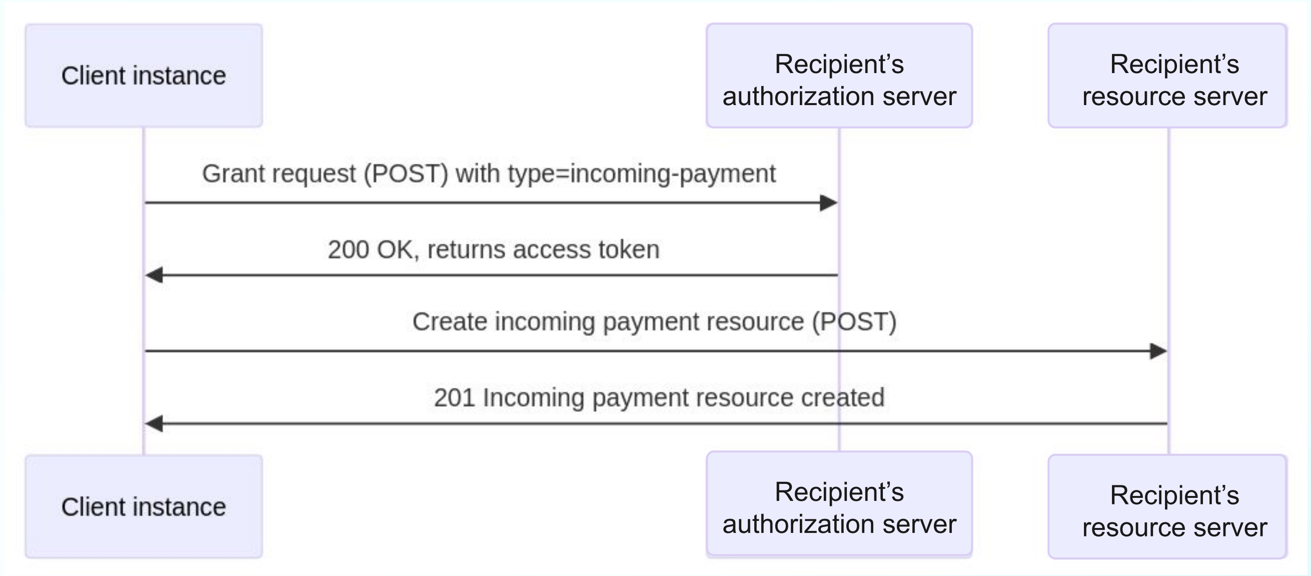
- Different configuration options:
  - One account can be defined by a single wallet address.
  - One account can have multiple wallet addresses pointing to it.

    Allowing account holders to generate unique wallet addresses for every client they interact with can help prevent a single address from becoming a tracking vector.
  - A single wallet address can also represent all of a user's accounts.

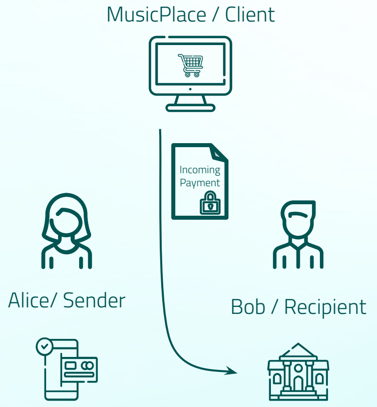
    This way the provider determines which one of the accounts should receive the payment based on the currency type or some other agreed upon criteria.

Open Payment Flows

Incoming payment



Client instance
Recipient's authorization server
Recipient's resource server
Grant request (POST) with type=incoming-payment
200 OK, returns access token
Create incoming payment resource (POST)
201 Incoming payment resource created
Client instance
Recipient's authorization server
Recipient's resource server



Open Payment Flows

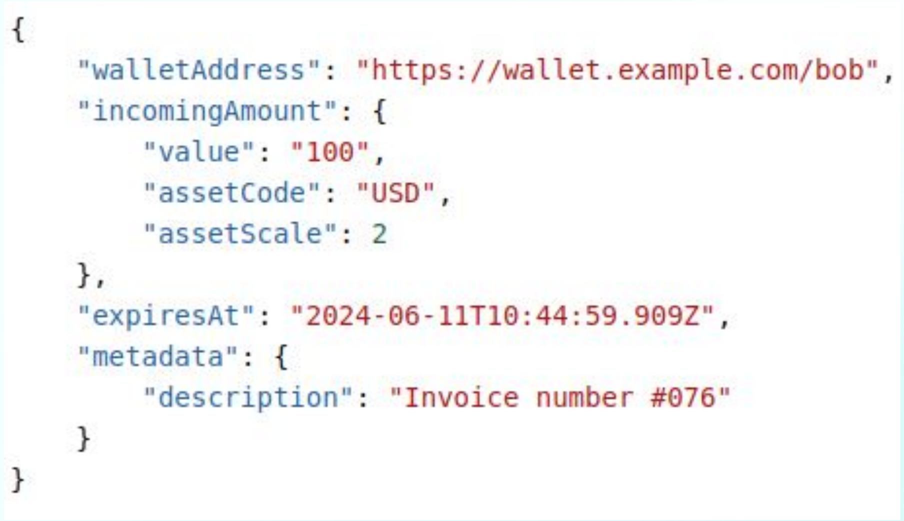
Incoming payment request

Specify the amount the recipient should receive

- ➔ Include an `incomingAmount` value

Specify the amount the sender is sending

- ➔ Excluding the `incomingAmount` in the incoming payment request
- ➔ Include a `debitAmount` of the amount you want to send within a Create Quote request (referred to as a fixed-send quote)



```
{
    "walletAddress": "https://wallet.example.com/bob",
    "incomingAmount": {
        "value": "100",
        "assetCode": "USD",
        "assetScale": 2
    },
    "expiresAt": "2024-06-11T10:44:59.909Z",
    "metadata": {
        "description": "Invoice number #076"
    }
}
```

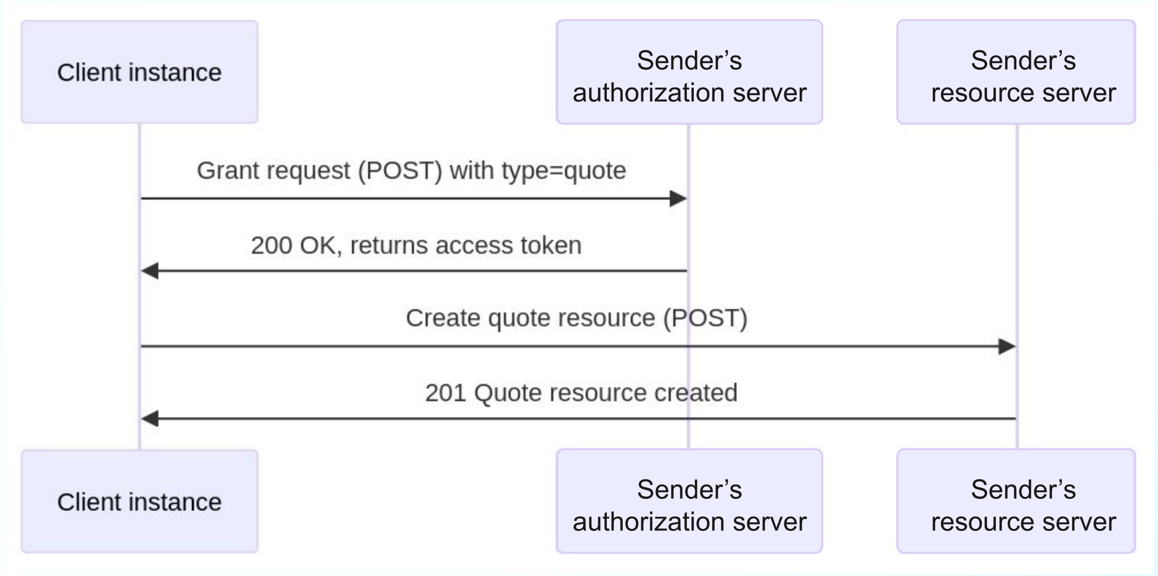
Open Payment Flows

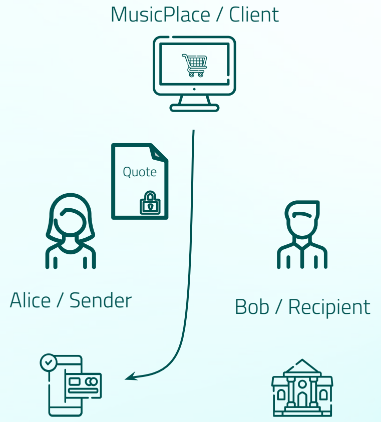
Incoming payment response

- When using ILP as a payment method in OP, the following information is required from the recipient's ASE, in the incoming payment's method object.
    - A type of ilp to indicate the payment method.
    - The wallet address of the recipient's ASE.
    - A shared secret: A cryptographically secured secret to be exchanged between the sender's ASE and the recipient's ASE to ensure that packets sent over the Interledger network through a STREAM connection can only be read by the two parties.

Open Payment Flows

Quote


Client instance
Sender's authorization server
Sender's resource server
Grant request (POST) with type=quote
200 OK, returns access token
Create quote resource (POST)
201 Quote resource created
Client instance
Sender's authorization server
Sender's resource server

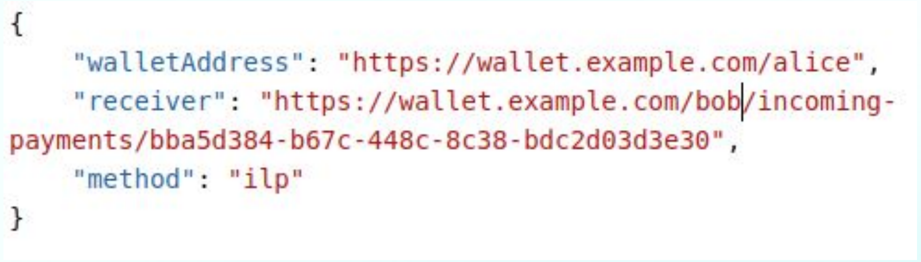


➔ The quote returns the amount it will cost to make the payment and a quote id in the form of a URL.

Open Payment Flows

Quote

- ➔ Fixed-receive quote - A fixed amount will be paid into the recipient's account. receiveAmount is required.
- ➔ Quote with defined incomingAmount - The receiver is the URL of the incoming-payment resource that will be paid into.
- ➔ Fixed-send quote - A fixed amount will be paid from the sender's account. debitAmount is required.

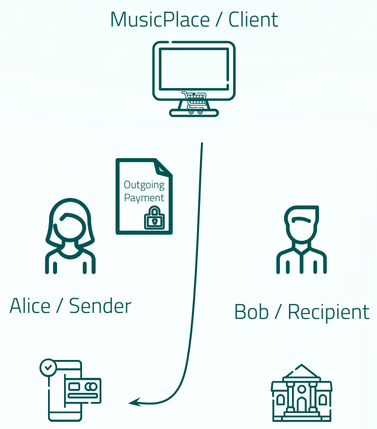


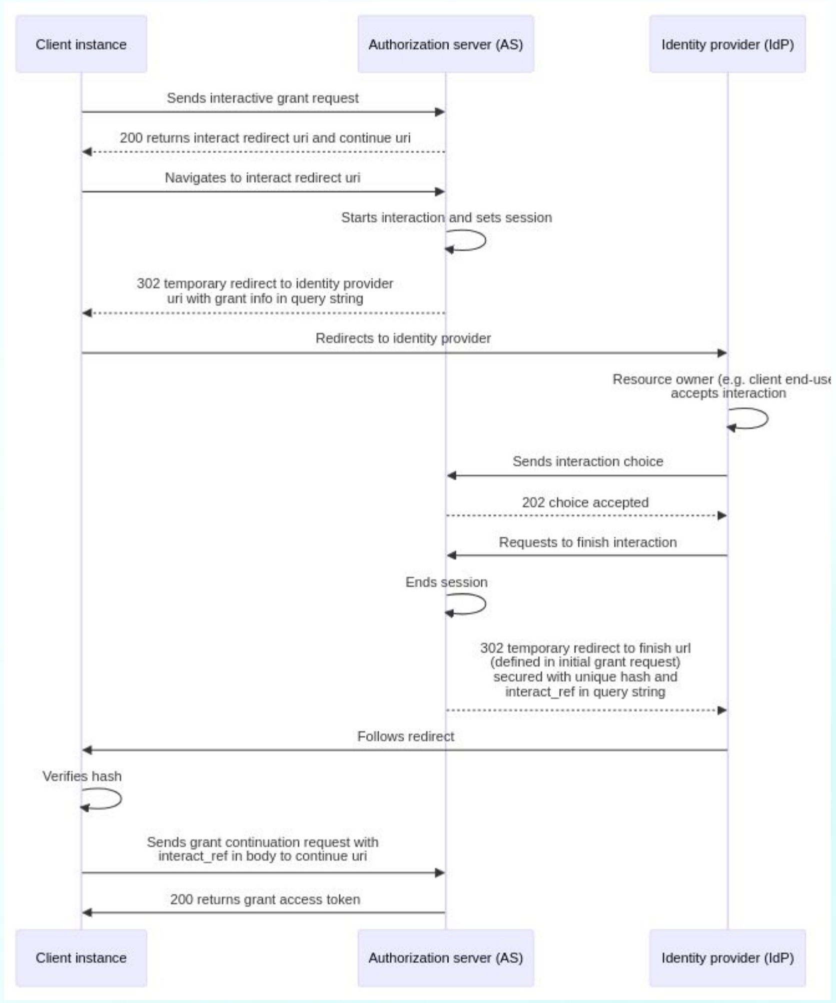
```
{
    "walletAddress": "https://wallet.example.com/alice",
    "receiver": "https://wallet.example.com/bob/incoming-
payments/bba5d384-b67c-448c-8c38-bdc2d03d3e30",
    "method": "ilp"
}
```

Open Payment Flows

Outgoing payment

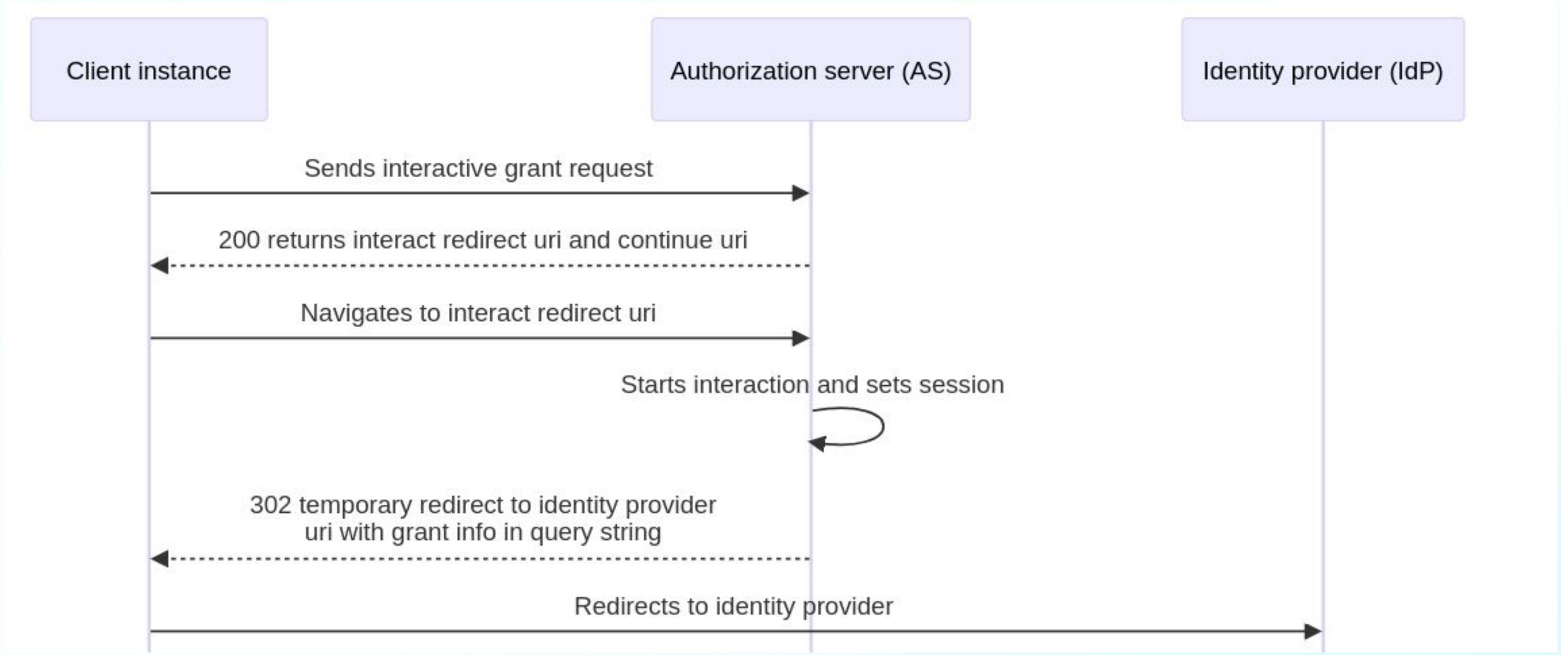
- ➔ Open Payments requires the client to send an interactive grant request to the authorization server before creating an outgoing payment resource.




Client instance
Authorization server (AS)
Identity provider (IdP)
Sends interactive grant request
200 returns interact redirect uri and continue uri
Navigates to interact redirect uri
Starts interaction and sets session
302 temporary redirect to identity provider uri with grant info in query string
Redirects to identity provider
Resource owner (e.g. client end-use accepts interaction
Sends interaction choice
202 choice accepted
Requests to finish interaction
Ends session
302 temporary redirect to finish url (defined in initial grant request) secured with unique hash and interact_ref in query string
Follows redirect
Verifies hash
Sends grant continuation request with interact_ref in body to continue uri
200 returns grant access token
Client instance
Authorization server (AS)
Identity provider (IdP)

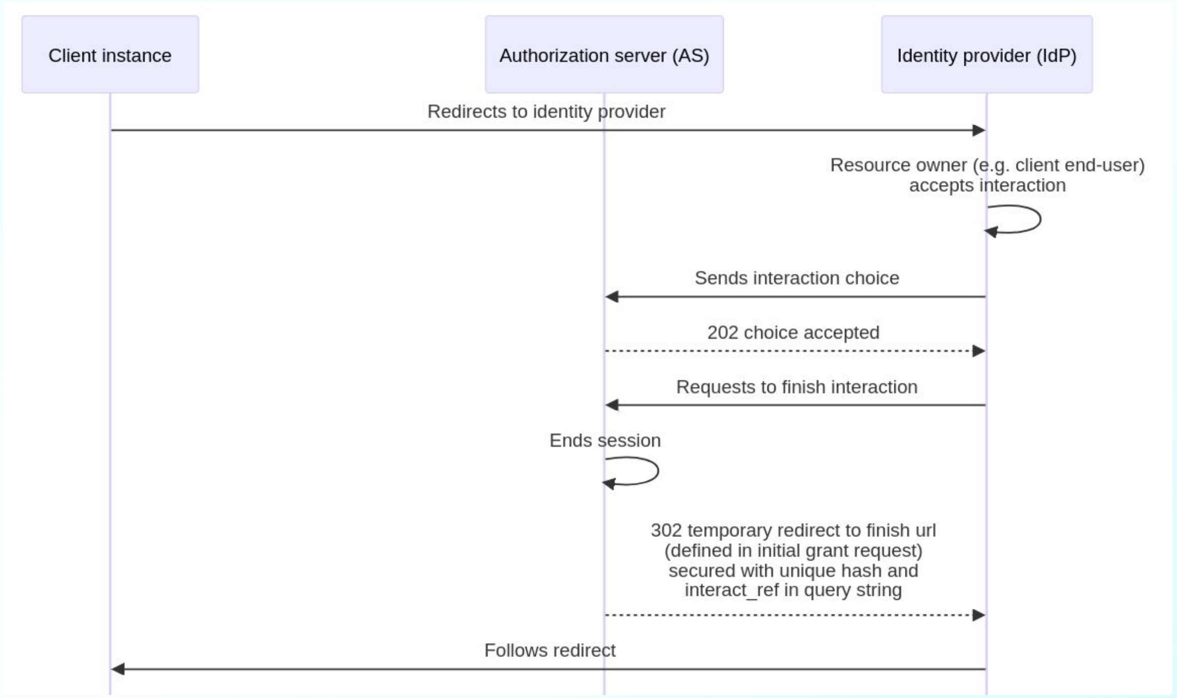
Open Payment Flows

Outgoing payment


Client instance
Authorization server (AS)
Identity provider (IdP)
Sends interactive grant request
200 returns interact redirect uri and continue uri
Navigates to interact redirect uri
Starts interaction and sets session
302 temporary redirect to identity provider uri with grant info in query string
Redirects to identity provider

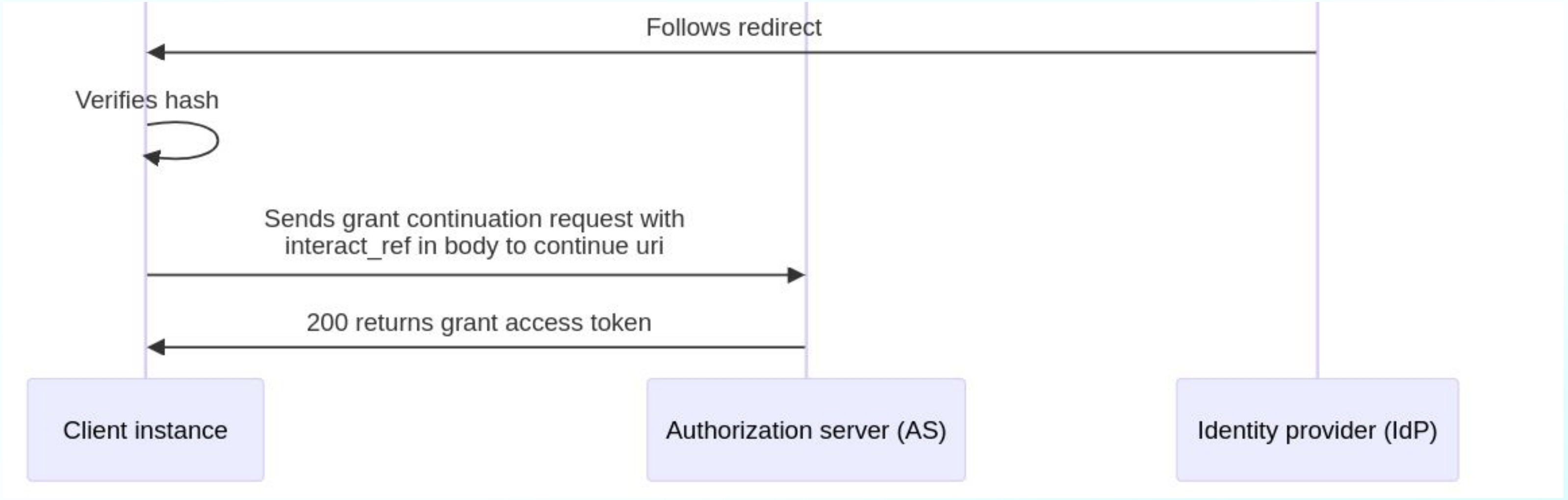
Open Payment Flows

Outgoing payment


Client instance
Authorization server (AS)
Identity provider (IdP)
Redirects to identity provider
Resource owner (e.g. client end-user) accepts interaction
Sends interaction choice
202 choice accepted
Requests to finish interaction
Ends session
302 temporary redirect to finish url (defined in initial grant request) secured with unique hash and interact_ref in query string
Follows redirect

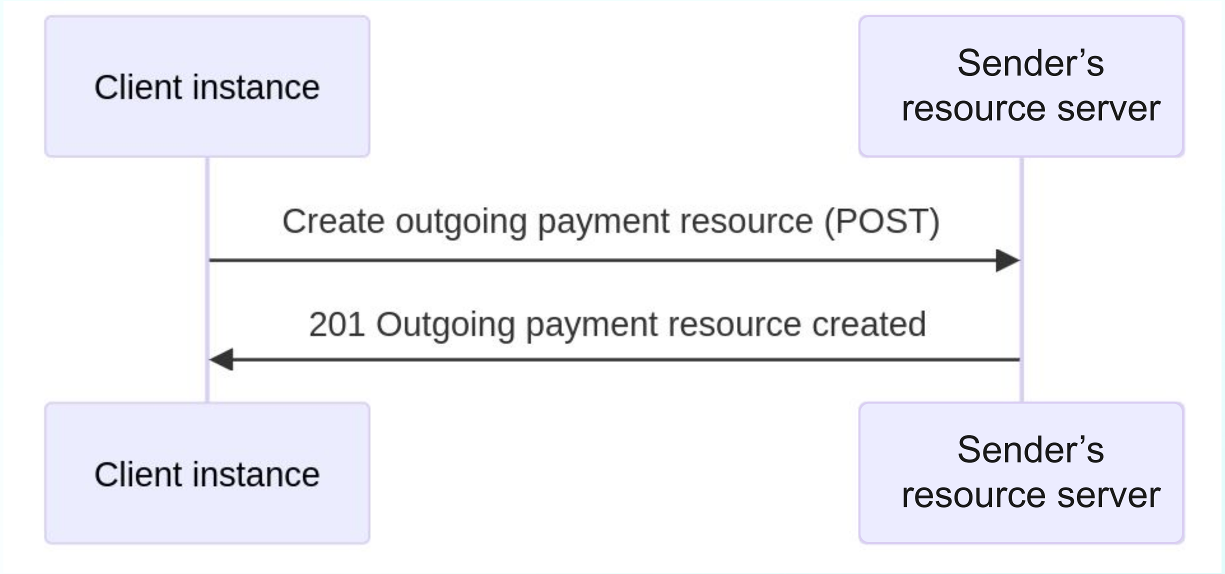
Open Payment Flows

Outgoing payment


Follows redirect
Verifies hash
Sends grant continuation request with interact_ref in body to continue uri
200 returns grant access token
Client instance
Authorization server (AS)
Identity provider (IdP)

Open Payment Flows

Outgoing payment



➔ After the outgoing-payment resource is created, the incoming payment can complete and the Open Payments flow ends.

Deep Dive: Authorization

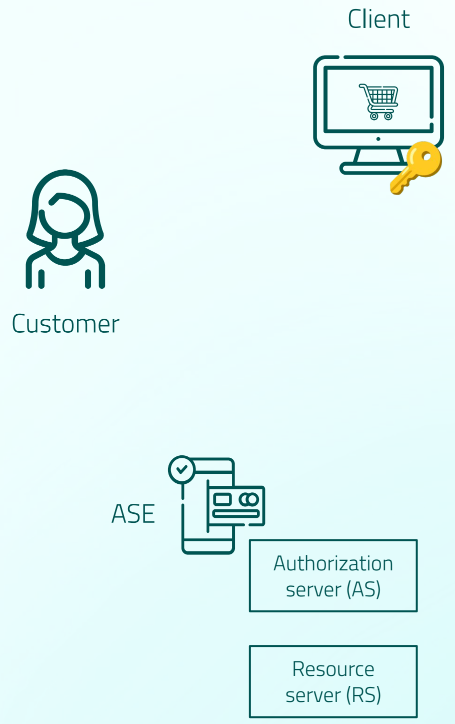
Authorization Protocols

- ➔ Giving a third party access to your resources
- ➔ E.g. OAuth and GNAP
- ➔ Authentication: User logs into a service (e.g. Google / authorization server).
- ➔ Authorization: User grants permission to the third-party app to access specific resources (e.g. apply a filter to my photo / make a payment from my account).
- ➔ Access Token: Service (Google / authorization server) provides the third-party app with an access token.
- ➔ Resource Access: The third-party app uses the token to access the user's resources from the service (Google photos / resource server).

GNAP

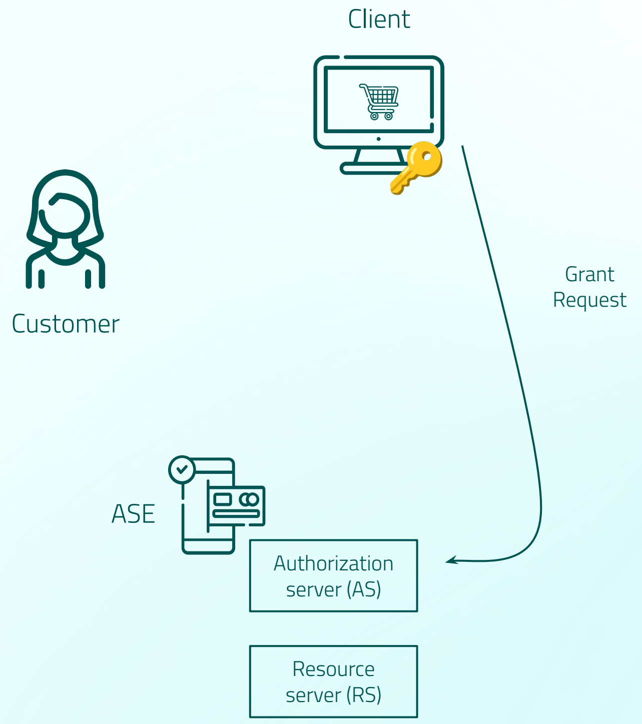
Grant **N**egotiation and **A**uthorization **P**rotocol

- ➔ Open Payments uses the Grant Negotiation and Authorization Protocol (GNAP) to give the client application authorization to use the Open Payments API to interface with a customer's account.
- ➔ An authorization server (AS) implements the GNAP standard in order to give clients access tokens.
- ➔ Clients use the access tokens to access resources on the resource server (RS), on the customer's behalf.



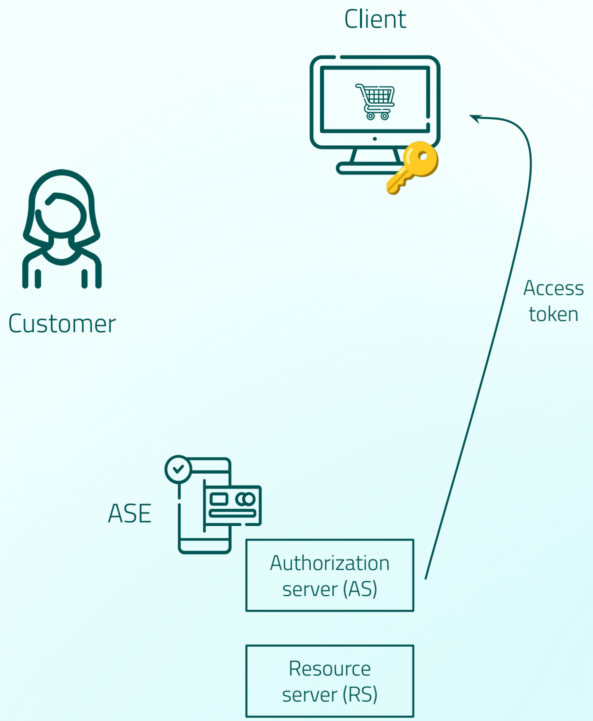
Grant Flows

- ➔ Before a client can access the Open Payments API, it must send a grant request to the AS.
- ➔ The request must contain the type of resource it wants to work with and the action(s) it wants to take on the resource.
- ➔ Resource types include incoming-payment, quote, and outgoing-payment.
- ➔ The available actions depend on type, but examples include create and read.



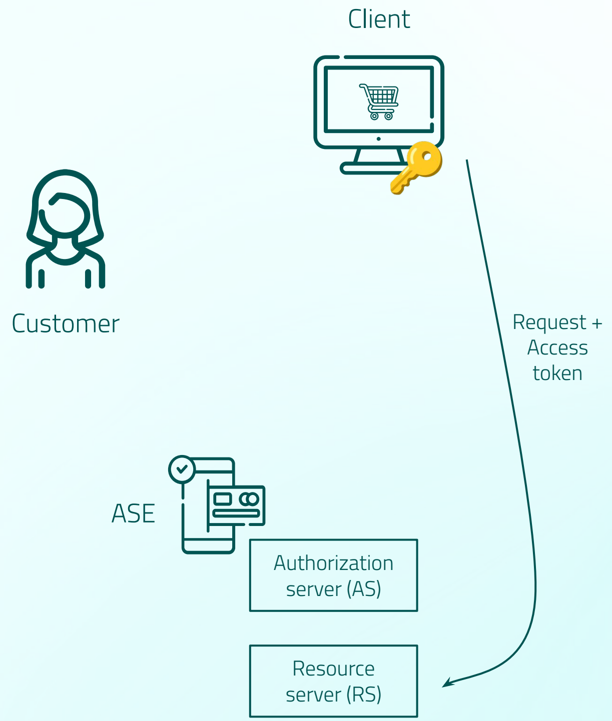
Grant Flows

- A successful grant request results in the AS returning one or more access tokens to the client.
- Access tokens represent a set of access rights and/or attributes granted to the client.



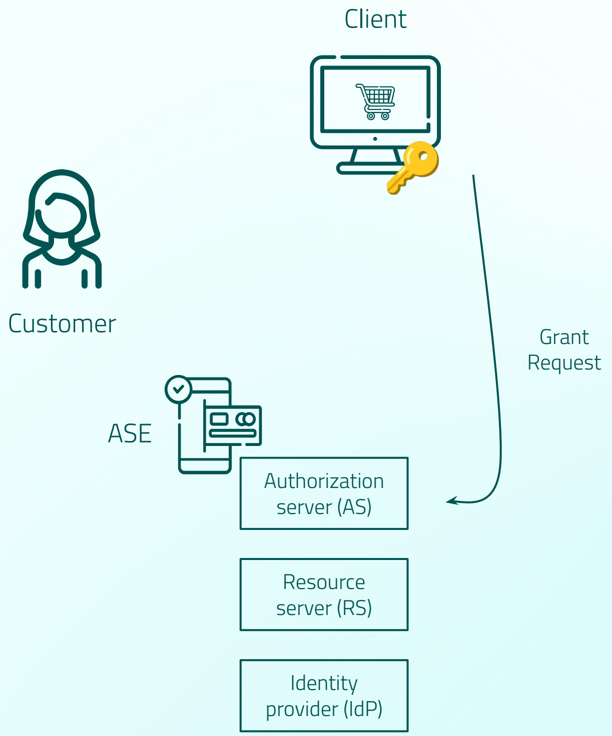
Grant Flows

- ➔ With the access token(s), the client can access a resource server's (RS) Open Payments API and perform authorized operations.
- ➔ E.g. creating incoming payments on behalf of the resource owner (the customer).



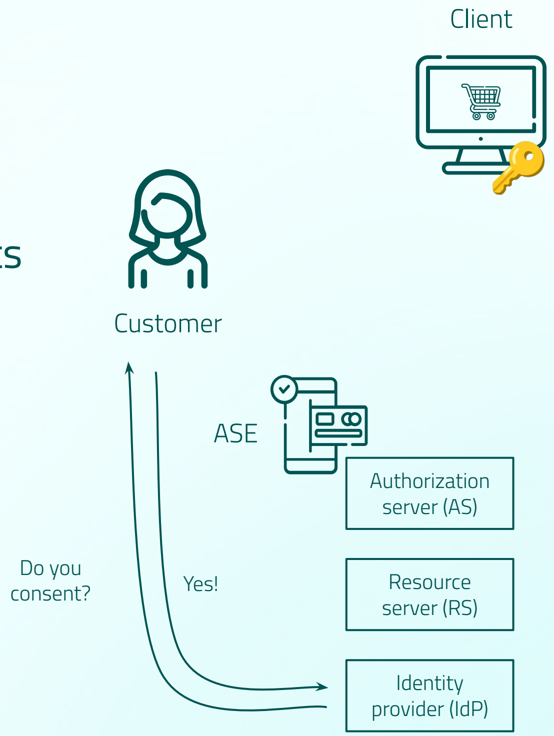
Interactive Grant Flows

➔ Upon receiving a grant request, the authorization server redirects the client to the identity provider.



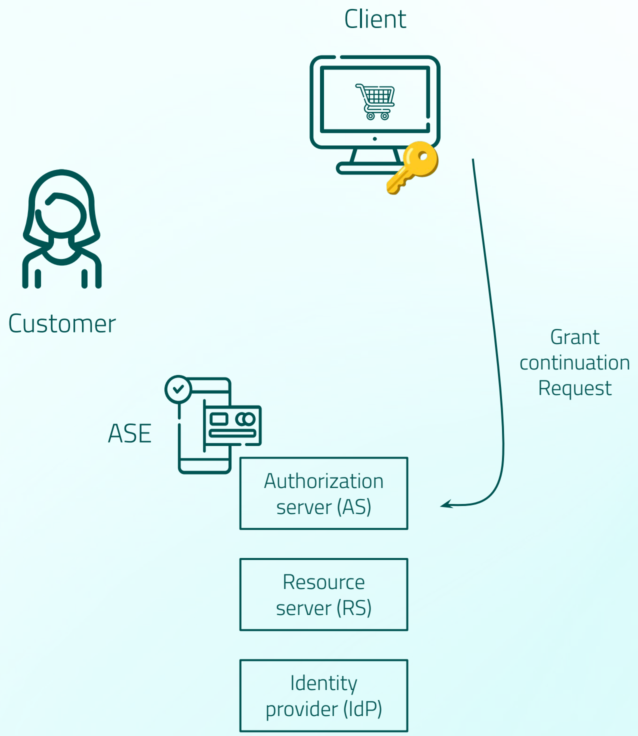
Interactive Grant Flows

- ➔ The identity provider must check that it's talking to the customer, then get the customer's consent.
- ➔ Once it has the customer's consent, it redirects back to the client.



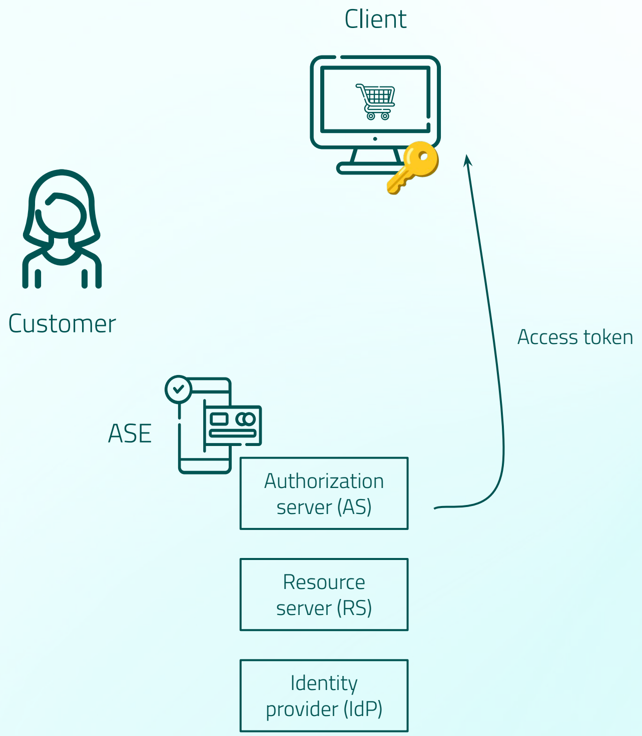
Interactive Grant Flows

➔ The client sends a grant continuation request to the AS.



Interactive Grant Flows

- The AS can then grant an access token to the client application.
- Now the client can continue making requests to the resource server (RS) with its access token(s).



Grants

The sequence of requests a client makes when setting up a payment can follow the path below.

1. Request an incoming-payment grant from the receiver’s AS
2. Send request to create an incoming payment resource to the receiver’s RS
3. Request a quote grant from the sender’s AS
4. Send request to create a quote resource to the sender’s RS
5. Request an interactive outgoing-payment grant from the sender’s AS
6. Send request to create an outgoing payment resource to the sender’s RS

Grants

- ➔ A grant indicates a transfer of authorization from a resource owner (RO) to a piece of software (the client application).
- ➔ Grant types match our resource types:
  - ◆ Incoming payment
  - ◆ Quote
  - ◆ Outgoing payment
- ➔ Grants serve as a request from the client to the authorization server. Once the grant is validated, it can be exchanged for access tokens.
- ➔ Grants are typically short-lived or single-use in many cases.

Access tokens

- ➔ An access token is issued by the authorization server.
- ➔ It is used by the client to access specific resources on the resource server.
- ➔ A client will produce the access token to a resource server to access protected resources.
- ➔ Access tokens can be short or long lived.

Security

- ➔ Successful grant requests result in access tokens.
- ➔ Access tokens ensure that only authorized operations are executed.
- ➔ We’re still left with two very important questions:

1. How do we know we can trust that the client sending the request is who they say they are?
2. How can we be sure that the request hasn’t been tampered with?

Client Keys

- ➔ All client requests in Open Payments are signed using a unique key that identifies the client to the authorization and resource servers.
- ➔ A client must generate and add its key to a key registry before requesting a grant for the first time. Each client is represented by its own wallet address.
- ➔ A key registry is a list of keys associated with client applications.
- ➔ Key registries are publicly exposed endpoints. Servers can retrieve the client's key registry by accessing WALLET_ADDRESS/jwks.json
- ➔ This allows servers to verify that a client is who they says they are.

Interlude: Asymmetric Keys

- Key Pair Generation: Two keys are created: one public and one private.
- Encryption & decryption: Data encrypted with one key can only be decrypted with the other key.
- Since the private key is kept secret, only its owner can encrypt / decrypt the data.
  - In our use case the private key is used for encryption and the public key decrypts.
- Security: Even though the public key is shared, the private key remains confidential.
- Integrity: Ensures that data cannot be tampered with.
- Authentication: Confirms the identity of the sender.

Interlude: Asymmetric Keys

In The Open Payments Context

- ➔ Each client generates a key pair: a public key and a private key.
- ➔ Public key is registered with the Account Servicing Entity (ASE) in a key registry.
- ➔ Clients sign their request using their private key.
- ➔ ASE verifies the signature using the client's public key that is associated with them in the key registry.
- ➔ This ensures the request is from the legitimate client and hasn't been tampered with.
- ➔ Non-repudiation: The client cannot deny sending the request.
- ➔ Purpose: integrity and authenticity, NOT privacy.

Interlude: Digital Signatures

Client Side: Creating The Signature

- Inputs:
  - Request data
  - Client's private key
- Process:
  - The request data is hashed (converted into a fixed-size string).
  - The hash is encrypted with the client's private key, creating the signature.
- Output:
  - The signature is a unique encrypted string.

Interlude: Digital Signatures

Server Side: Verifying The Signature

- ➔ Inputs:
  - ◆ Received request data
  - ◆ Signature from the request
  - ◆ Client's public key
- ➔ Process:
  - ◆ The server hashes the received request data.
  - ◆ The server decrypts the signature using the client’s public key to get the original hash.
  - ◆ The server compares both hashes. If the hashes match, the signature is valid.

Interlude: HTTP Requests

- ➔ An HTTP request is a message sent by a client to a server (in our case either the AS or RS) to request data or perform an action.
- ➔ HTTP Header:
  - ◆ Metadata about the request or response.
  - ◆ Provides essential information like content type, status codes, authentication credentials, etc.
- ➔ HTTP Body:
  - ◆ The actual data being sent in the request or response.
  - ◆ Contains the content, such as form data, JSON payload, or HTML.

Client Requests

➔ It’s important for a client to provide a way to consistently identify itself across requests to the AS. As such, clients must include the following:

- Header
  - Signature-Input: that includes the keyId associated with the client’s key pair
  - Signature: generated based on the Signature-Input, using the EdDSA signing algorithm (as proof of the private key the client owns)
- Body
  - Client: property containing the client’s wallet address

What makes GNAP a good fit?

Gives end users (customers) fine-grained control over:

- ➔ What resources they are granting access to
  - ◆ Incoming payments / quotes / outgoing payments
- ➔ What kinds of access they are granting
  - ◆ Read / write / update / delete
- ➔ How often they are granting this access
  - ◆ Once off / repeated / monthly / weekly

Zooming Out Again

The bigger picture

- ➔ Interledger is about sending instructions not sending money.
- ➔ The Interledger Protocol suite gives us
  - ◆ Standardisation
  - ◆ Low transaction costs
  - ◆ Scalable solution
  - ◆ Potential to revolutionise payment experiences
  - ◆ Enables micro transactions
- ➔ OP sits on top of ILP and gives us
  - ◆ Easy ways for third parties to get direct access to customer accounts.