ExPay: an exercise of a payment API

Overview

ExPay is an exercise of a RESTful payment API.

The API provides fetch (GET), list (GET without id), create (POST), update (PUT) and delete (DELETE) RESTful operations for payments. (ref: HTTP Methods)

Install

```
go get h12.io/expay
make test
go install h12.io/expay/cmd/expay
expay -h
# expay -host [host] -storage [storage]
```

Code layout

```
expay/ all domain types and constants
    cmd/ contain all main packages of services
        expay/ expay service main package
    db/boltdb a boltdb implementation of expay.DB interface
    service/ contain logic of all services
        payment/ payment service logic
    testdata/  data for testing
```

Storage

Multiple storage backends could be supported given the following interface:

```
    DB interface {
        Create(v interface{}) (id string, err error)
        Get(id string, v interface{}) error
        Delete(id string) error
        Update(id string, v interface{}) error
        List() (Iter, error)
    }
    Iter interface {
        Next() bool
        Scan(v interface{}) (id string, err error)
        Close() error
    }
```

Two backends are currently supported:

- boltdb: ACID persistent KV store
- fakeDB: a memory based DB for unit testing

API Document

- SwaggerHub: https://app.swaggerhub.com/apis/h12w/expay-api/1.0.0
- PDF API Documents (see below if in PDF)

ExPay API

Overview

ExPay API provides a RESTful payment API

Version information

Version : 1.0.0

Paths

Create payment

```
POST /v1/payments
```

Description

This will create a new payment

Parameters

Type	Name	Description	Schema
Body	**payment** *optional*	Payment info	Payment

Responses

HTTP Code	Description	Schema
201	PaymentResponse is an envelope for a payment response	PaymentResponse
400	ErrorResponse is returned when an error occurred	ErrorResponse
500	ErrorResponse is returned when an error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

List payments

```
GET /v1/payments
```

Description

This will show all available payments

Responses

HTTP Code	Description	Schema
200	PaymentResponse is an envelope for a payment response	PaymentResponse
500	ErrorResponse is returned when an error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

Update payment

```
PUT /v1/payments/{id}
```

Description

This will update the payment with the ID

Parameters

Type	Name	Description	Schema
Path	**id** *required*	ID is payment ID	string
Body	**payment** *optional*	Payment info	Payment

Responses

HTTP Code	Description	Schema
200	PaymentResponse is an envelope for a payment response	PaymentResponse
400	ErrorResponse is returned when an error occurred	ErrorResponse
500	ErrorResponse is returned when an error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

Delete payment

```
DELETE /v1/payments/{id}
```

Description

This will delete the payment with the ID

Parameters

Type	Name	Description	Schema
Path	**id** *required*	ID is payment ID	string

Responses

HTTP Code	Description	Schema
200	PaymentResponse is an envelope for a payment response	PaymentResponse
500	ErrorResponse is returned when an error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

Definitions

BeneficiaryParty

BeneficiaryParty represents the beneficiary party of a payment

Name	Schema
account_name *optional*	string
account_number *optional*	string
account_number_code *optional*	string
account_type *optional*	integer (int64)
address *optional*	string
bank_id *optional*	string
bank_id_code *optional*	string
name *optional*	string

Charge

Charge contains the amount and currency of a change

Name	Schema
amount *optional*	string

Name	Schema
currency *optional*	string

ChargesInformation

ChargesInformation contains changes information of a payment

Name	Schema
bearer_code *optional*	string
receiver_charges_amount *optional*	string
receiver_charges_currency *optional*	string
sender_charges *optional*	< Charge > array

DebtorParty

DebtorParty represents the debtor party of the payment

Name	Schema
account_name *optional*	string
account_number *optional*	string
account_number_code *optional*	string
address *optional*	string
bank_id *optional*	string

Name	Schema
bank_id_code *optional*	string
name *optional*	string

ErrorResponse

ErrorResponse is returned when an error occurred

Name	Description	Schema
code *optional*	status code	integer (int64)
message *optional*	error message	string

Fx

Fx of a payment

Name	Schema
contract_reference *optional*	string
exchange_rate *optional*	string
original_amount *optional*	string
original_currency *optional*	string

Links

Links of a payment response

Name	Description	Schema
self *optional*	self link	string

Payment

Payment represents a payment resource

Name	Schema
attributes *optional*	PaymentAttributes
id *optional*	string
organisation_id *optional*	string
type *optional*	string
version *optional*	integer (int64)

PaymentAttributes

PaymentAttributes contains properties of a payment

Name	Schema
amount *optional*	string
beneficiary_party *optional*	BeneficiaryParty
charges_information *optional*	ChargesInformation
currency *optional*	string

Name	Schema
debtor_party *optional*	DebtorParty
end_to_end_reference *optional*	string
fx *optional*	Fx
numeric_reference *optional*	string
payment_id *optional*	string
payment_purpose *optional*	string
payment_scheme *optional*	string
payment_type *optional*	string
processing_date *optional*	string
reference *optional*	string
scheme_payment_sub_type *optional*	string
scheme_payment_type *optional*	string
sponsor_party *optional*	SponsorParty

PaymentResponse

PaymentResponse is an envelope for a payment response

Name	Description	Schema
data *optional*	an array of payment	< Payment > array
links *optional*		Links

SponsorParty

SponsorParty represents the sponsor party of a payment

Name	Schema
account_number *optional*	string
bank_id *optional*	string
bank_id_code *optional*	string