Form3 Coding Exercise Design

Assumptions

- The API paths contain the plural form of resources, and the HTTP method defines the kind of action to be performed on the resource.
- {paymentId} is assumed to be universally unique (UUID).
- I have assumed 'List a collection of payment resources' to mean 'Fetch a collection of payment resources'.

API

/v1/public/payments/{paymentId}

HTTP Method

GET

Description

Fetch a payment resource.

Comments

{paymentId} should match the PaymentData::id field in the PaymentData structure detailed below.

Return Status

- 200 Ok
- 404 Not Found
- 400 Bad Request
- /v1/public/payments

/v1/public/payments

HTTP Method

POST

Description

Create a payment resource.

Comments

A new payment will be created with the POST data, which should be JSON representation of the `PaymentData` structure detailed below.

Return Status

- 201 Created
- 400 Bad Request

/v1/public/payments/{paymentId}

HTTP Method

PUT

Description

Update a payment resource.

Comments

- {paymentId} should match the PaymentData::id field detailed in the PaymentData structure below.
- The payment data associated with {paymentId} will be replaced with the PUT data, which should be JSON representation of the PaymentData structure detailed below.
- This API is distinct from POST (above), and will fail with 404 if {paymentId} is not found.

Return Status

- 200 Ok
- 404 Not Found
- 400 Bad Request

/v1/public/payments/{paymentId}

HTTP Method

DELETE

Description

Delete a payment resource.

Comments

- {paymentId} should match the PaymentData::id field detailed in the PaymentData structure below.
- DELETE on collection not supported since:
    - asymmetric with respect to single-item DELETE
    - dangerous operation

Return Status

- 204 No Content
- 404 Not Found
- 400 Bad Request

/v1/public/payments

HTTP Method

GET

Description

Fetch a collection of payment resources (please see Assumptions).

Comments

The parameters offset and limit from `PaymentDataContainer` can be specified to constrain the requested payments.

Return Status

- 200 Ok
- 409 invalid limits
- 400 Bad Request

Data Structures

The following details the JSON data structures for payments, passed along with calls to:

- /v1/public/payments (POST) - Create a payment resource (`PaymentData sent`)
- /v1/public/payments/{paymentId} (PUT) - Update a payment resource (`PaymentData sent`)

The following JSON is received by calls to:

- /v1/public/payments/{paymentId} (GET) - Fetch a payment resource (`PaymentDataWrapper received`)
- /v1/public/payments (GET) – Fetch a collection of payment resources (`PaymentDataWrapper received`)

```
PaymentDataWrapper {
        code (int, optional): The HTTP status code of the returned result
        status (string, optional): A string description of the call status
        copyright (string, optional): The copyright notice for the returned result
        etag (string, optional): A digest value of the content returned by the call
        results (PaymentDataContainer, optional): The results returned by the call
}
```

```
PaymentDataContainer {
        offset (int, optional): The requested offset (number of skipped results) of the call
        limit (int, optional): The requested result limit
        total (int, optional): The total number of resources available given the current filter set
        count (int, optional): The total number of results returned by this call
        data (Array[PaymentData], optional): The list of payments returned by the call
}
PaymentData {
        type (string, optional): The type of data (should be "Payment" for this result)
        id (string, optional): UUID for this data
        version (int, optional): Version used
        organisation_id (string, optional): UUID for the organisation
        attributes (PaymentAttributes, optional):
}
PaymentBeneficiaryParty {
        account_name (string, optional)
        account_number (string, optional)
        account_number_code (string, optional)
        account_type (int, optional)
        address (string, optional)
        bank_id (string, optional)
        bank_id_code (string, optional)
        name (string, optional)
}
PaymentChargesInformation {
        bearer_code (string, optional)
        sender_charges (Array[PaymentSenderCharges], optional)
        receiver_charges_amount (string, optional)
        receiver_charges_currency (string, optional)
}
PaymentSenderCharges {
        amount (string, optional)
        currency (string, optional)
}
PaymentDebtorParty {
        account_name (string, optional)
        account_number (string, optional)
        account_number_code (string, optional)
        address (string, optional)
        bank_id (string, optional)
        bank_id_code (string, optional)
```

```
        name (string, optional)
}
PaymentFx {
        contract_reference (string, optional)
        exchange_rate (string, optional)
        original_amount (string, optional)
        original_currency (string, optional)
}
PaymentSponsorParty {
        account_number (string, optional)
        bank_id (string, optional)
        bank_id_code (string, optional)
}
PaymentAttributes {
        amount (string, required)
        beneficiary_party (PaymentBeneficiaryParty, optional)
        charges_information (PaymentChargesInformation, optional)
        currency (string, optional)
        debtor_party (PaymentDebtorParty, optional)
        end_to_end_reference (string, optional)
        fx (PaymentFx, optional)
        numeric_reference (string, optional)
        payment_id (string, optional)
        payment_purpose (string, optional)
        payment_scheme (string, optional)
        payment_type (string, optional)
        processing_date (string, optional)
        reference (string, optional)
        scheme_payment_sub_type (string, optional)
        scheme_payment_type (string, optional)
        sponsor_party (PaymentSponsorParty, optional)
}
```