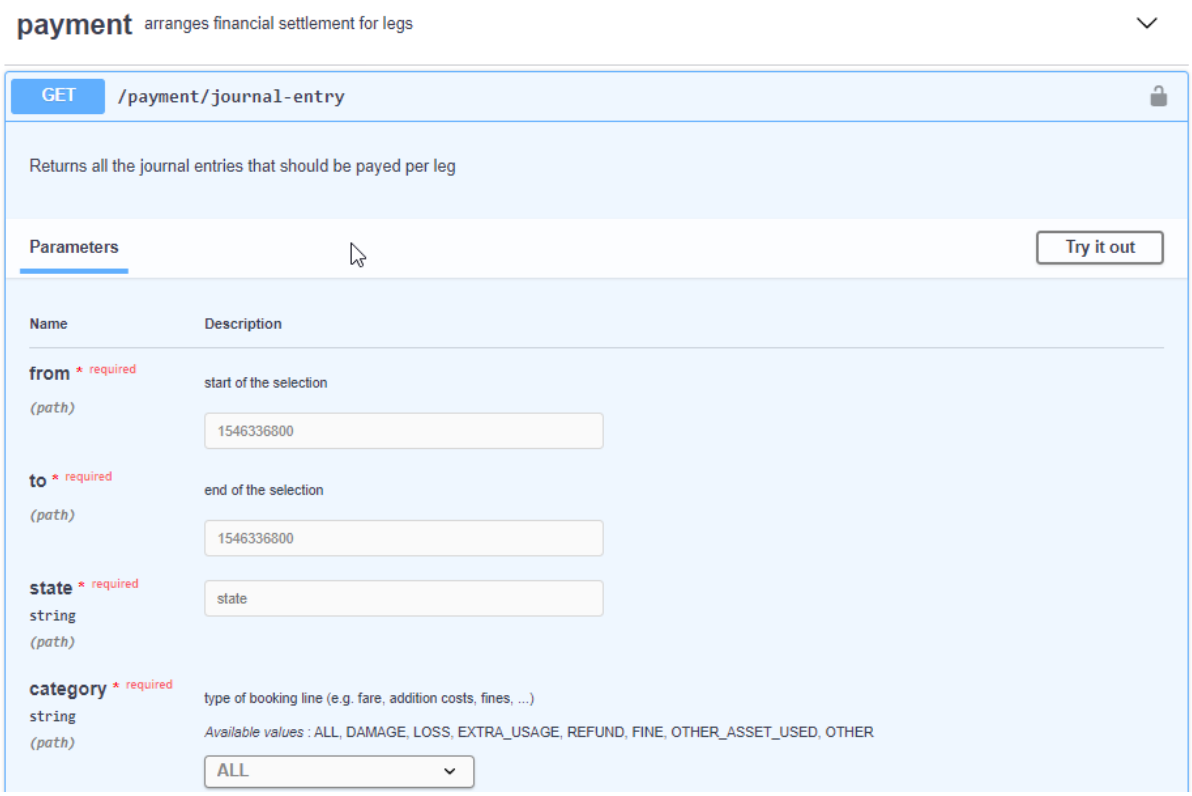
#75 endpoints payment, #78 fines etc, #74 'Asset now'

I bundled these issues, they have a technical overlap. The payments I've put in an object called 'journal-line'. The journal lines can be a fare or describe extra costs. These journal-lines can be requested by as well the TO as the MSP per period, and optionally a type (FINE, DAMAGE, LOSS,..). One single leg can have multiple journal lines, e.g. for the fare and later on one for a fine.

Main properties of the journal line are state and expiration-date; the amount of money depends on the type of journal line. If it's a fare, it will be the sum of all parts (fixed and flex parts).

Technical

75: added endpoint
78: added extra-costs in details journal-line
74: extended fare object. The MSP and TO should end the trip at the finish of the leg, thereby fixating the fare. The TO should change the fare. 2 scenario's: a) TO finishes the leg, the MSP should request the booking again with updated fare. B) the MSP finishes the leg, the updated fare should be in the response. Also added a condition ('deposit-condition', see return conditions).



Returns journal-entry[]

journal-entry:
id:
external-url:
state:
enum: [OPEN, PAYED, DISCUSS]
expiration-date:
comment:
details:

```
      oneOf:
        - $ref: '#/components/schemas/fare'
        - $ref: '#/components/schemas/extra-costs'

  extra-costs:
    allOf:
    - $ref: '#/components/schemas/amount-of-money'
      reason:
        enum: [DAMAGE, LOSS, EXTRA_USAGE, REFUND, FINE, OTHER_ASSET_USED, OTHER]
      description:
      meta:
        type: array
        items:
          $ref: '#/components/schemas/key-value'

  fare:
    description: the total fare is the sum of all parts.
    properties:
      parts:
        type: array
        items:
          $ref: '#/components/schemas/fare-part'

  fare-part:
    description: this describes a part of the fare (or discount). It contains a for instance the startup
costs (fixed) or the flex part (e.g. 1.25 EUR per 2.0 MILES). The amount is tax included. In case of
discounts, the values are negative.

    example: { "amount": 9.96, "currency-code": "EUR", "tax-rate": 21.0, "type": "FLEX", "unit-type":
"HOUR", "units": 1 }

    allOf:
      - $ref: '#/components/schemas/amount-of-money'
    properties:
      type:
        enum: [FIXED, FLEX]
      unit-type:
        description: in case of 'FLEX' mandatory. E.g. 0.5 EUR per HOUR
        enum: [KM, MINUTE, HOUR, MILE]
      units:
        type: number

  amount-of-money:
    type: object
    properties:
      amount:
        description: This should be in the base unit as defined by the ISO 4217 currency code with the
appropriate number of decimal places and omitting the currency symbol. e.g. if the price is in US
Dollars the price would be 9.95
```

```
      currency-code:
        description: ISO 4217 currency code

      tax-rate:
        description: tax rate (percentage of amount)
        example: 21.0
```

#61 asset return conditions

To facilitate the flexible conditions around assets (different return area's per asset instead of return area's per TO), we've added 'conditions' in the planning-options object. It's an array of conditions with a key. In the result you can refer to these keys per asset. This way you can tell the MSP that an asset has a deposit of 20 euro and has to be returned at a specific location between opening hours.

Planning-option {
"conditions": [{ "name": "DEPOSIT_20", "conditionType": "DEPOSIT", "amount": 20.0, "currency-code": "EUR", "tax-rate": 21.0 }, { "conditionType": "RETURN-AREA", "name": "Alkmaar oost", "opening-times": […] }],
"results": [{ …, "conditions": ["DEPOSIT_20", "Alkmaar oost"], …}]
}

Technical

The return conditions can be different per asset of the same TO. It's not static data. Therefore the 'conditions' are added in the planning module:

```
  planning-options:
    properties:
      conditions:
        type: array
        items:
          $ref: '#/components/schemas/condition'

      results:
        type: array
        items:
          type: object
          properties:
...
            conditions:
              description: references to the 'conditions' array (start of this object).
              type: array
              items:
                type: string

  condition:
    type: object
    oneOf:
      - $ref: '#/components/schemas/postponed-commit-condition' (supports postponed-commit, #72)
      - $ref: '#/components/schemas/return-area-condition'
      - $ref: '#/components/schemas/deposit-condition'  (supports pay-as-you-go, #74)
    properties:
```

```
    name:
      type: string

  return-area-condition:
    properties:
      conditionType:
        type: string
        example: 'RETURN-AREA'
      name:
        type: string
      station-id:
        description: optional station id (see static information)
        type: string
      geometry:
        $ref: '#/components/schemas/polygon'
      opening-times:
        description: the opening times of the facility
        type: array

        items:
          $ref: '#/components/schemas/period'

  deposit-condition:
    properties:
      conditionType:
        example: 'DEPOSIT'
    allOf:
      - $ref: '#/components/schemas/amount-of-money'
```

#72 postponed commit

To facilitate this scenario, I've added an extra state (CONDITIONAL-CONFIRMED) can be set be the TO to inform that it's not yet completely confirmed. Whenever the subcontractor confirms, the commit event can be fired to the MSP to make the state CONFIRMED. Otherwise the newly added DENY event can be fired. The state will become cancelled after that.
In the planning stage the MSP should be informed that this can happen with the returned asset (his process should be handled differently). Therefore the postponed-commit-condition is introduced. In this condition is also communicated how long the state 'CONDITIONAL-CONFIRMED' can exist. In the time has expired, it will become a booking in EXPIRED state.

Technical

For the postponed commit there is added an extra state in the booking process:

```
  booking-state:
    description: The life-cycle state of the booking (from NEW to FINISHED)
    type: string
    enum: [NEW, PENDING, RELEASED, EXPIRED, CONDITIONAL-CONFIRMED, CONFIRMED,
CANCELLED, STARTED, FINISHED]
```

And is added a new event to go from 'conditional-confirmed' to 'cancelled' : DENY. To go from 'conditional-confirmed' to 'confirmed' the 'COMMIT' event will be used.

```
  booking-operation:
    properties:
      operation:
        enum: [CANCEL, EXPIRE, DENY, COMMIT]
```

The duration of the state ‘conditional-confirmed’ must be communicated in the conditions.

```
  postponed-commit-condition:
    properties:
      conditionType:
        type: string
        example: 'POSTPONED-COMMIT'
      ultimate-response-time:
        $ref: '#/components/schemas/timestamp'
```

#81 more details passengers

The planning stage needs more information about the travelers. For instance age (influences the fare), abilities (influences the offer of assets), licenses and cards.

Technical

Was:

planning-check:

…

```
      travellers:
        description: the amount of people that have to travel from `from` to `to`
[https://github.com/efel85/TOMP-API/issues/56]
        type: number
```

…

Added:

planning-check:

++++

```
      users:
        type: array
        items:
          $ref: '#/components/schemas/user'

  user:
    type: object
    properties:
      age:
        type: number
      licenses:
        type: array
        items:
          $ref: '#/components/schemas/license'
      cards:
```

```
      type: array
      items:
        $ref: '#/components/schemas/card'
    requirements:
      type: array
      items:
        $ref: '#/components/schemas/key-value'

  license:
    type: object
    properties:
      country:
        $ref: '#/components/schemas/country'
      asset-type:
        $ref: '#/components/schemas/asset-class'

  card:
    type: object
    properties:
      card-type:
        type: string
      country:
        $ref: '#/components/schemas/country'
      asset-type:
        $ref: '#/components/schemas/asset-class'
```

#77 discount

The discount is added in the fare object. All original fare prices should be in there, but the sum of all parts is the total price. Therefore the discount should be added in the fare object as negative fare-parts.