Combination of Open Journey Planner OJPFareRequest with TOMP-API

Research Paper

Status	Draft
Version	0.9
Date of last modification	May 10, 2024
Authors	Diogo Ferreira (MENTZ), Matthias Günter (SBB), Markus Meier (SBB)
License	Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

Change History

Version	Status	Changes	by	Date
0.1	Draft	Initial version	D. Ferreira, M. Günter	2023-06-07
0.8	Draft	Added abstract, glossary, introduction, motivation, formatting etc.	M. Meier	2023-08-16
0.8.1	Draft	Findings S. de Konink	M. Meier	2023-09-12
0.9	Release	Findings by D. Ferreira	M. Meier	2024-05-10

Abstract

Modern trip planners have powerful capabilities to compute end-to-end trip itineraries, including public and private services. Based on preloaded service data (timetables, service times, etc.), they compute reliable trip plans. However, these trip plans will often not include reliable, binding information on fares (prices).

When travellers then want to proceed to booking, payment and usage (actual travel), they will typically want to see fares first, e.g. for a taxi ride, before committing to a given trip plan and move on to booking.

In terms of API standards, the OJP standard should be used for trip planning; for booking, usage and payment, TOMP-API or OSDM could become the dominant standards in the future. In between, we see a need for APIs/services for fare information, based on the combination of OJPFareRequest and TOMP-API /planning/inquiries endpoint.

This study investigates how OJPFareRequest and TOMP-API /planning/inquiries can be mapped back and forth, so that fare information can be integrated into the use case sequence for trip planning, booking, travelling and paying.

We conclude that a mapping, and thus coexistence of the two standards (APIs) is possible, but some minor, non-breaking modifications will be required, which we will propose as CRs to the respective boards.

In a future companion paper, we will investigate the mapping of the proposed OJP 2.0 OJPAvailabilityRequest on TOMP-API.

Table of Contents

1 Glossary with Terms, Abbreviations and References

Abbreviation	Full name, explanation and hyperlinks
CR	Change Request; a proposal for a change of a given standard (new version) that would facilitate its adoption. We intend to propose, or have already proposed, the CRs to the boards responsible for the respective standard.
FOT	Swiss Federal Office of Transport, https://www.bav.admin.ch/bav/en/home.html
MaaS	Mobility as a Service, https://en.wikipedia.org/wiki/Mobility_as_a_service.
MP	Mobility (Service) Provider, aka MaaS Provider. Typically, the provider of an end-customer-facing app or webapp for mobility services (MaaS). In the following use cases and processes, the MP acts as the client which triggers the processes.
OJP	Open Journey Planner. In our context, OJP has a dual meaning: - the term is commonly used to refer to the CEN standard Open API for Distributed Journey Planning, https://www.transmodel-cen.eu/ojp-standard. GitHub: https://github.com/VDVde/OJP/issues. - a Swiss trip planner funded by FTO, in service since 2022, based on the OJP standard, https://opentransportdata.swiss/en/dataset/ojp2020.
OJPFare, OJPFare Request	A part of the OJP standard, regarding fare information. On GitHub: - https://github.com/VDVde/OJP/blob/master/OJP_Requests.xsd - https://github.com/VDVde/OJP/blob/master/OJP_Fare.xsd.
OSDM	Open Sales and Distribution Model, a standard aimed at booking processes of rail and other public transport systems, https://osdm.io.
SBB	Swiss Federal Railways, largest Swiss railway operator, https://www.sbb.ch/en.
SC	Service Catalogue, aka Service Directory or Service Registry. A system for looking-up URLs and meta-data of service endpoint APIs of the TOs.
SKI	System Tasks in Customer Information, a department at SBB tasked for data exchange for public transport customer information, https://oev-info.ch.
SKI+	A team at SKI supporting FOT in building a national data infrastructure for mobility (both public and private), opentransportdata.swiss/en.
TO	Transport Operator, a company offering transport services through an API. The services may be produced either by the TO's own infrastructure (vehicles), or by aggregating and reselling transport services of other TOs. At the extreme, a TO can be a broker, i.e., a comprehensive, large aggregator offering the services of many other TOs.
TOMP-API	Transport Operator to Mobility-as-a-service Provider API, an open standard for booking of mobility services, https://tomp-wg.org. GitHub: https://github.com/TOMP-WG/TOMP-API, API documentation: https://app.swaggerhub.com/apis/TOMP-API-WG/transport-operator_maas_provider_api/1.5.0.

2 Introduction

Trip planning is a fundamental service or functionality needed by travellers who want to travel using public and/or private transport services instead of an own vehicle (car, bike). The OJP standard defines a European (CEN) API standard for trip planning. In Switzerland, an equally named OJP service has been introduced in 2022, funded by FOT.

For a subsequent booking, payment and usage of transport services, OJP provides no support; other APIs such as TOMP-API or OSDM are needed.

Thus, if providers of trip-planner apps (smartphone or webapps) want to integrate booking, payment and usage functionality – in other words, provide MaaS functionality –, they will have to switch from the OJP standard (for planning) to TOMP-API or OSDM standards (for booking).

In this study, we investigate this switch of standards from OJP to TOMP-API in detail:

- We uncover a hiatus (aka gap) between trip planning with OJP and TOMP-API, being the lack of fare information for on-demand or private trip legs.
- We then propose a bridge for this gap, based on the combination of OJPFareRequest and TOMP-API /planning/inquiries, and their respective mapping.

Furthermore, we believe that bridging this gap will be possible and provide a building brick for the construction of viable MaaS systems.

2.1 The Problem: A Hiatus between Planning and Booking

2.1.1 Trip Planners

A trip planner service in a smartphone app or webapp allows travellers to plan trips with public and private transport operators end-to-end (A to B). The service will combine trip legs of both public and private transport services in a smart way, so that the traveller reaches the destination B efficiently and based on some personal preferences such as convenience or price.

A trip planner algorithm performs, casually speaking, some shortest-path calculation in a network of possible paths (transport services), to minimise parameters such as travelled distance or the time spent. For this purpose, large volumes of data (public transport timetables, road maps, real-time updates, etc.) are imported and pre-processed beforehand by the trip-planner.

An example of such a service, based on the OJP standard, is the equally named Swiss OJP system funded by FOT, in service since 2022. OJP includes the entire Swiss public transport, but also some private services such as taxis, on-demand buses or micro-mobility.

Over the years, trip planners have become more and more comprehensive and powerful. However, for reasons of performance and scaling, these trip planners typically cannot include third-party API calls for availability or pricing information. Featuring all public transport information of an entire country, the response time of the journey planner is in the order of hundreds of milliseconds, while a single API request over the network towards one external system might already cost the same. A journey planner would aim for complete integration of all availability to include or exclude the option. As a consequence, response times would grow enormously.

2.1.2 OJP standard for trip planner APIs

The OJP standard was developed as a standard for trip planning APIs. The first major version 1.0 defines the basic trip planning service (OJPTripRequest, trip from A to B), plus extra services for exchange points, fares, multipoint trips, place information, stop events, and trip information.

OJP 1.0 does not support the subsequent steps of availability, booking request, booking, trip execution, payment, and fulfilment. For these subsequent steps, different API standards need to be used, such as TOMP-API or OSDM.

The release of the new major version OJP 2.0 is expected in summer 2024. OJPAvailablityRequest will be added to provide service-availability information. This may be seen as a step in the direction of booking services. Currently (as of spring 2024), however, there are no plans to actually add booking functionality to OJP.

Thus, for booking, a different standard needs to be used. The main candidates for this purpose are, in our opinion, the TOMP-API and OSDM standards. In this study, we investigate the TOMP-API variant.

2.1.3 Booking with TOMP-API

TOMP-API, in contrast to OJP, aims at the subsequent booking. It provides two closely related endpoints:

- **/planning/inquiries** provides a non-binding, informative offer that may include fare information, but which is not mandatory[1].
  This endpoint should have short response times and tolerate heavy load (many requests).
- **/planning/offers** provides a binding, bookable offer with a booking id[2]. Depending on the internal planning system of the service provider, a call to this service could already trigger reservation and allocation of resources (e.g., a vehicle or driver) for the leg and keep the reservations alive for a given time (e.g. 2 minutes). "fare" is not a mandatory attribute in the given data structure, but in a reasonable implementation, it will have to be provided, otherwise travellers will not want to book an offer.
  Due to these extra functionalities (reservation, resource allocation and fare calculation), this endpoint will typically be slower and not suitable for heavy load.

2.1.4 The Hiatus: Lack of Fare information in OJPTripRequest (OJPTripDelivery)

Thus, the hiatus between OJP trip planning and TOMP-API booking is about fares: The OJPTripDelivery will typically not include reliable fare information for private legs, but before proceeding to booking of a leg with TOMP-API endpoints, the travellers will want to see reliable, binding fare information first.

2.2 The Bridge: OJPFareRequest and TOMP-API /planning/inquiries

The solution approach here is to integrate fare information in the trip planning where needed, typically when the travellers navigate to the "details views" in their trip planner apps.

Fare information can be obtained from corresponding API services of the transport operators. Within the OJP standard, OJPFareRequest is the standard of choice for this.

However, a private operator, e.g. a taxi broker or a sharing operator, may be providing TOMP-APIs only. As discussed above, to allow for short response times and heavy load, the normal implementation would be with TOMP-API /planning/inquries.

[1] In Switzerland, an implementation of /planning/inquries which also returns an indicative price is strongly suggested for mobility providers.

[2] An offer id will be passed to the client system with OJP 2.0 .

We strongly recommend that /planning/inquiries should include the fare information, otherwise API clients who need fares might be forced to use the /planning/offers endpoint. In the remainder of this study, our focus is on /planning/inquiries, assuming that it does include fares.

Thus, the mapping between OJPFareRequest and TOMP-API /planning/inqurіes may become a necessity, or, in other words, the bridge across the gap.

2.3 The Focus of this Study

In this study, we thus focus on the OJP Fare service (OJPFareRequest) for travel-fare information, as a key to bridging the gap between the preceding trip planning step and the subsequent booking steps. With the fare service, trip plans can be completed with fare information, so that the travellers feel confident to commit to a trip plan and move on to booking.

We focus on the interaction of the respective standards:

- **OJPFareRequest**, which, as a member of the OJP standards family, fits well in combination with OJPTripRequest.
- **TOMP-API /planning/inquiries** endpoint, which is intended to provide fast information (non-binding offers), including fare information as well.

In the future, both OJPFareRequest and TOMP-API /planning/inquiries may be used in parallel, and in some use case scenarios, there may be a need for adapters or converters mapping from one to the other.

Thus, we investigate the mapping between these two standards, both requests and responses, and provide some illustrative examples for both.

2.4 Use Case Sequence

Our study subject is the use case sequence of trip planning, fare calculation and booking:

1. **Trip planning** (OJPTripRequest) comprises the traveller's steps to find a suitable trip itinerary from A to B.
2. **Fare calculation** (OJPFareRequest) will provide important fare information to the trip plans so that the traveller can decide for one given trip plan.
3. **Booking**: To commit and actually travel based on the trip plan, booking APIs (TOMP-API /planning/offers and /bookings) must be called with the right parameters.

In more detail, the use case sequence comprises these steps:

1. **Trip planning** A to B with OJPTripRequest, presenting a handful of proposals for trip itineraries to the traveller. Each itinerary consists of one or several trip legs with various public or private transport operators. The trip plan must include all relevant travel information such as transport operators, times and locations for boarding and alighting, and routes.
2. **Fare calculation**: When requested by the traveller (e.g., when opening the details view on an itinerary), fare calculation for the respective legs should be triggered. This requires separate API calls for all legs where a fare is applicable – usually, all legs of type TimedLeg and of type ContinousLeg that are not an “own vehicle mode”. These steps consist of:
    a. Look-up of the OJPFare service endpoint in the service catalogue for a given leg.
    b. If it exists, call the endpoint to obtain the applicable fare.
    c. Integrate the fare details into the trip plan of step 1 and display them to the traveller.
3. When the traveller decides for a trip plan (all legs, or parts of it), e.g. by clicking a “book now” button, continue with the booking API calls. This requires a check-and-confirm step

of the traveller per leg, and will typically proceed leg by leg, starting with leg 1, etc. In case of a TOMP-API:

a. Look-up of the endpoints for a given leg in the service catalogue.
b. Call /planning/offers endpoint of the operator to get a binding offer (including a price/fare which is ideally the same as in step 2 but may differ). Display the offer to the traveller.
c. When the traveller confirms, e.g., by clicking on a “buy now” button, call /booking endpoint.

2.5 The Aspect of Flexibility / Late Commit

Many travellers will prefer to remain flexible as much as possible. They will want the trip planner to give them precise and reliable information on possible itineraries for a trip end to end, but often, they will not want to really commit to a plan end-to-end.

They will prefer to commit themselves to a given trip leg, e.g., a taxi ride, only shortly beforehand, when they are sure they can actually make it.

Thus, trip plans must be precise and comprehensive, but the traveller may want to book only the first part of it and do a replanning later.

In other words, the respective systems must allow for single and multi leg bookings.

2.6 Error Handling

Several errors and exceptions may occur:

- Fare or booking endpoints may be unavailable.
- A trip plan may have proposed a trip itinerary with some (hypothetical) legs, e.g., with a taxi service, but the respective calls to the fare or booking endpoint then fails (no fare information available, no binding offer available, etc.)

The traveller must be informed accordingly and will typically start over with a fresh trip plan for the full trip A to B, or part of it.

2.7 Possible Partitioning and Sequence of the Use Case

As discussed above, the use case sequence of planning, fare calculation and booking comprises a number of steps, some of which iteratively, and with various actors.

Various variants may occur and are depicted in the following sequence diagrams.

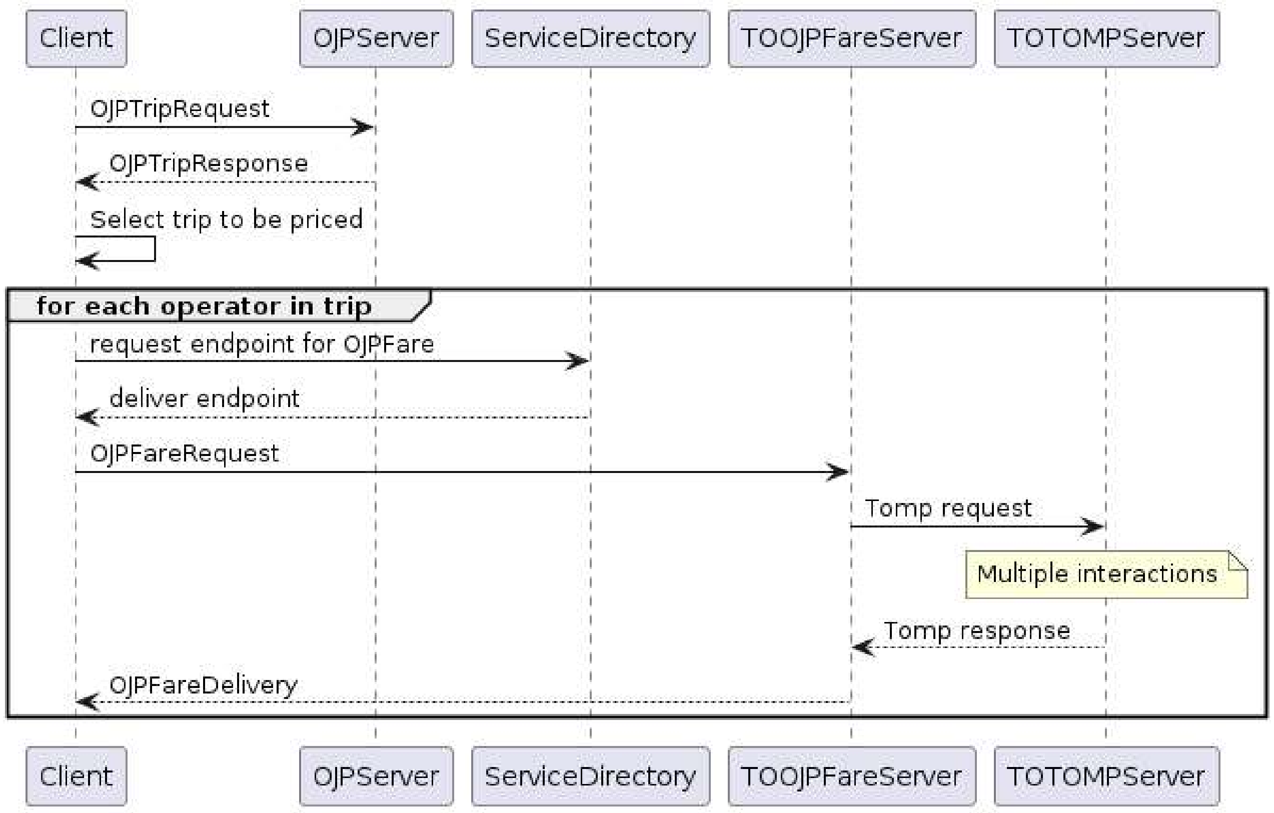

Figure 1: In variant 1, the client (traveller's app, MaaS app) does all the orchestration work.

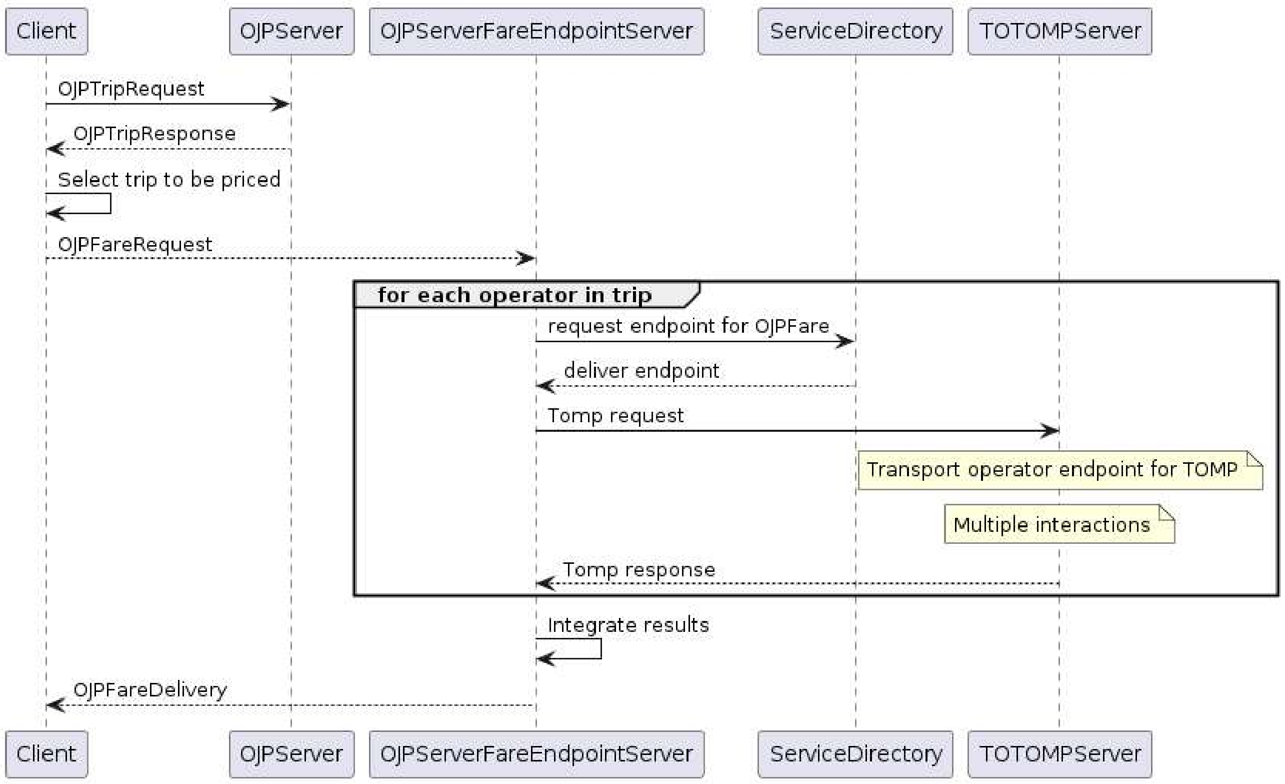

Figure 2: In variant 2, an endpoint by a TO does the orchestration work.

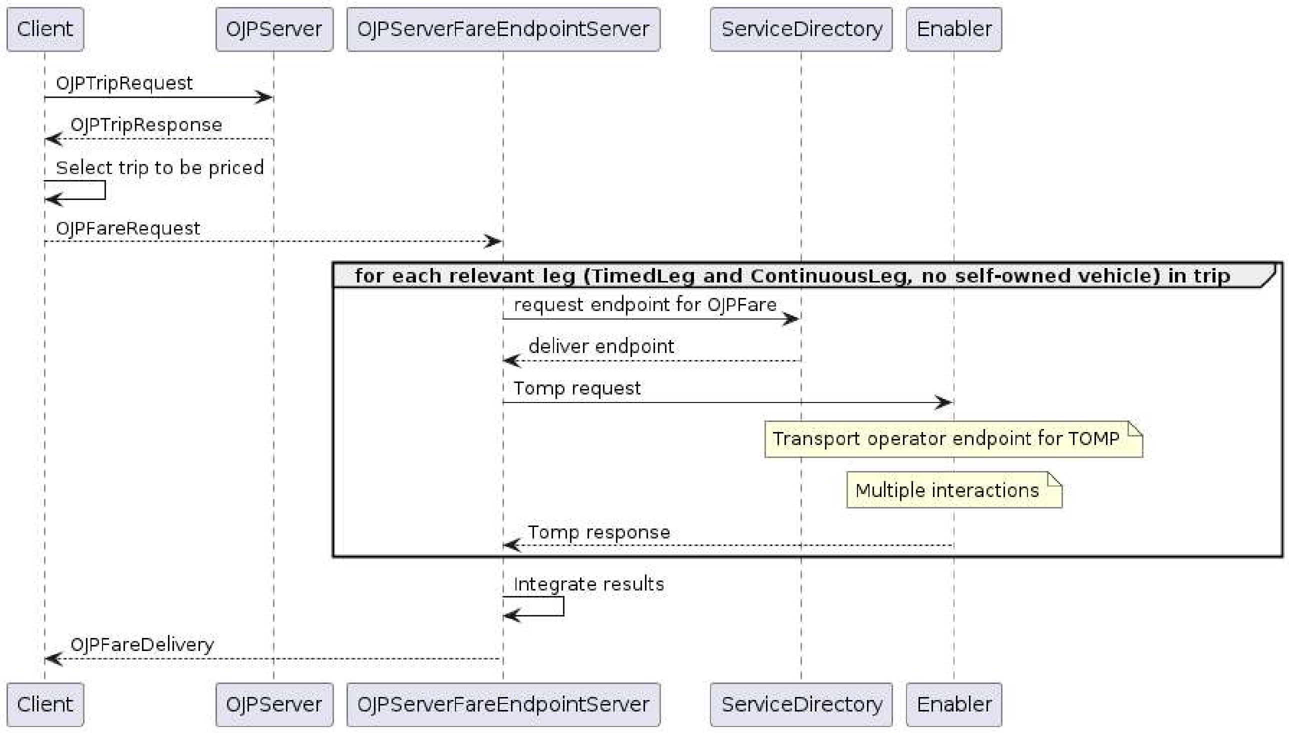

Figure 3: In variant 3, a central enabler does the transformation and orchestration work.

3 Mapping OJPFareRequest to TOMP-API /planning/inquiries

After trip planning, some of the legs of a given trip itinerary need to be booked. In the chain of exchange standards, this means a change from the OJP standard to TOMP-API or another standard for distribution, such as OSDM, as illustrated in figure 4.

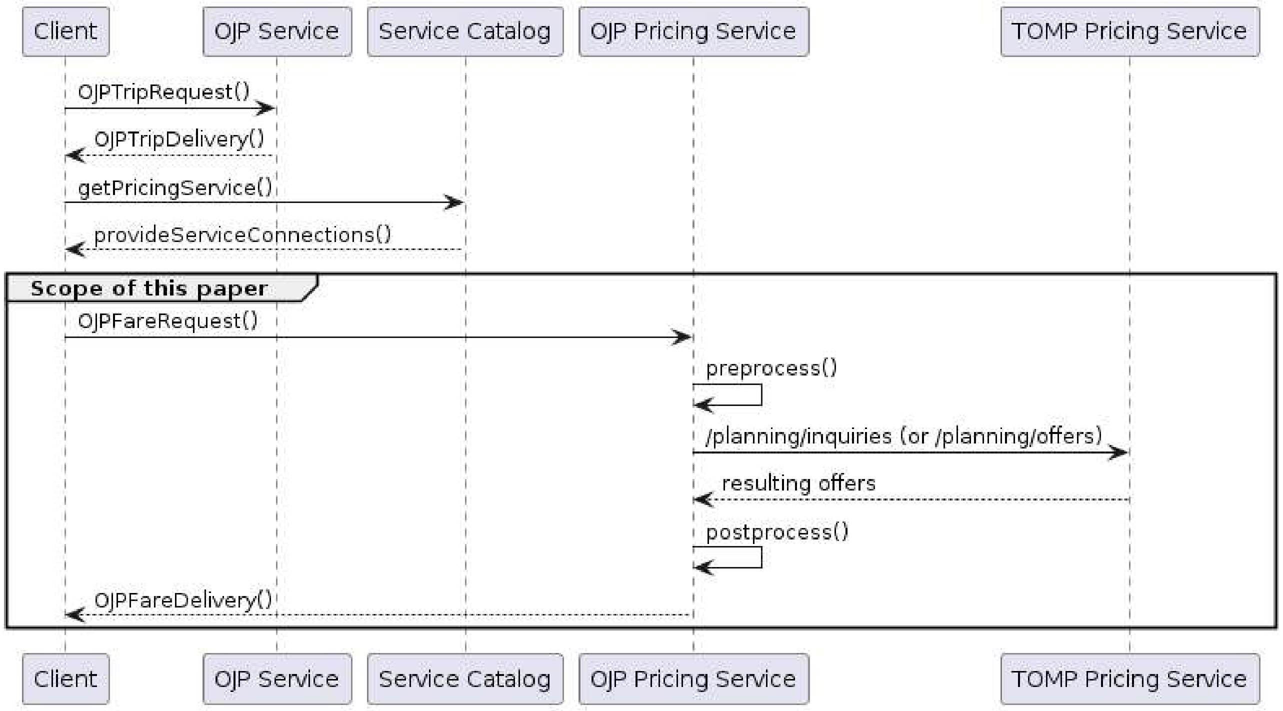

Figure 4: Sequence diagram of the relevant parts of the process[3]*.*

The following sections 3.1 and 3.2 describe the values which can be used in an OJPFareRequest (request and response) and the corresponding mapping on elements in a TOMP-API /planning/inquiries request and response.

3.1 OJPFareRequest → TOMP-API /planning/inquiries Request

OJPFareRequest has four different sub-variants: StopFareRequest, StaticFareRequest, TripFareRequest and MultiTripFareRequest. We investigate the mapping with TripFareRequest here.

For more details, please refer to the API documentations and GitHub hyperlinks in the glossary.

Table 1: Mapping from OJP TripFareRequest to TOMP-API /planning/inquiries request.

OJP TripFareRequest - Request	**TOMP-API /planning/inquiries - Request**	**Remarks**
OJP.FareResponse.FareResult.TripFareResult.ErrorMessage.Code	HTTP status code	OK

[3] For more details on the OJP and TOMP-API standard in general, please refer to the documentation hyperlinks in the introduction.

OJP.FareResponse.FareResult.TripFareResult.ErrorMessage.Text.Text	HTTP status message	OK
OJP.FareResponse.FareResult.TripFareResult.ErrorMessage.Text.TextId	n/a	OK
OJP.OJPFareRequest.TriPFareRequest.Trip TripStructure	from/to	Structure of the trip that is used to build the from/to for each TOMP-API request. Be aware that this typically will be split into multiple TOMP-API requests, one per leg. In TOMP-API, the request only provides "from" and to". However, namely for public transport, the whole trip (structure) would be better. In Switzerland, the pricing service (NOVA) needs all intermediate stops as well. Currently, we could add the TripStructure as an extension in extraInfo in the Swiss Profile for TOMP-API, but a general, standard solution would be better.

OJP.OJPFareRequest.TriPFare-Request.TripLeg.LegStart/Leg-End.Choic • StopPointRef (NMToken): the logical stop element (may be on the level of a STOP PLACE or QUAY) • StopPlaceRef (normal-izedString): The physical stop element • GeoPosition (siri:Location-Structure): The coordinates as WGS84 The following Types should be ig-nored in OJPFare in Switzerland: • TopographicPlaceRef • PointOfInterestRef • AddressRef	from/to.stopReference.type ENUM: • GTFS_STOP_ID • GTFS_STOP_CODE • GTFS_AREA_ID • CHB_STOP_PLACE _CODE • CHB_QUAY_CODE • NS_CODE from/to.place.stopReference.id: ex-amples: • NL:S:13121110 • BE:S:79640040 from/to.place.stopReference.coun-try (two-letter country codes ac-cording to ISO 3166-1). Example: NLCH	StopReference Object in TOMP-API is optional. If the data is used in OJPFare, it should be filled out. The Swiss TOMP-API Profile intends to always provide the co-ordinates. 1. Documentation on TOMP-API: "reference to a stop (can be nation specific). This can help to specific pinpoint a (bus) stop. Extra information about the stop is not supplied; you should find it elsewhere." 2. The ENUMs of stopRefer-enceType are not explained. We had to make some as-sumptions. *Mapping:* • *StopPointRef → CHB_QUAY_CODE* • *StopPlaceRef → CHB_STOP_PLACE_ CODE* • *Currently the DIDOK is used for the stopReference.id* ⚠CR TOMP-API: add a docu-mentation for the ENUMs. ⚠CR TOMP-API: changing this enumeration altogether?
OJP.OJPFareRequest.TripFare-Request.TripLeg.Service.Opera-torRefs	n/a	The OperatorRefs are the keys for the lookup of endpoints in the service catalogue. The must be used to get the end point. If an end point serves for several operators, the query parameter addressed-to should be used. ⚠CR TOMP-API: operators could be added for systems that serve more than one operator.

n/a	from/to.stationId	Documentation on TOMP-API: "reference to operator/stations". The stations describe the pickup spots of the vehicles (e.g. eScooter stations). The id is not mandatory in TOMP-API, though. Coordinates are and should do the job instead. See remarks there (next row). Not needed.

Possibilities to get the coordi-nates *
 	
1. OJP.OJPFareRequest. TripFareRequest.Tri-pLeg.LegStart/LegEnd.Call-Place.GeoPosition.Longitude
2. Client uses OJP.LocationInfor-mationRequestto get GeoLo-cation
3. Server is able to search for GeoPosition using the Refer-ences (StopPointRef, Stop-PlaceRef, TopographicPlaceRef, Poin-tOfInterestRef, AddressRef)
	from/*to*.coordinated.lng
	
	from/*to*.place.coordinated.lat
		No coordinate information in OJPFares if the CallPlace is not of type GeoPosition. Perhaps from the TripContext or obtained by us-ing OJP Location Information ser-vice.
		
		1. The client should always send coordinates to a TOMP-API system.
		2. It is therefore crucial that this information can be added at some point.
		3. Using stopReference.type and stopReference.id might be needed nevertheless for opti-mal response.
		
		⚠CR OJP: OJP should consider adding TripContext to OJPFare so that doing LIR in the converter is not needed.
		
		Remark: FareResponseContext added in OJP 2.0.

n/a	from/to.place.coordinates.alt	No altitude element in OJP. Possible use-cases: • Operator wants to use altitude difference from start to stop point for fare price calculation (example: eScooter uses more energy to get up a hill). • Navigation/routing in buildings with multiple levels/storeys. Not supported.
n/a	from/to.place.physicalAddress	Mapping AddressRef in OJP to phyiscalAddress in TOMP-API is hardly possible, as AddressRef is a normalizedString, while physi-calAddress is a structure. Not supported.

n/a	radius	TOMP-API doc.: "*Maximum distance in meters a user wants to travel to reach the travel option.*" Is only of interest to define a trip. When asking for a fare, the trip has already been defined. Not needed with OJPFares. Not supported.
OJP.OJPFareRequest.TripFareRequest.Trip[n].Distance OJP.OJPFareRequest.TripFareRequest.Trip.TimedLeg.LegTrack.TrackSection.Length	estimatedDistance	Documentation for estimatedDistance on TOMP-API: "*instead of using the from/to construct, it is also possible to give an indication of the distance to travel. The process identifier 'USE_ESTIMATED_DISTANCE' is used to indicate this scenario. Also in meters*". In OJP, this means that per leg, a LegTrack.TracSection must exist with the element LENGTH. If an operator wants to create a fare price for a distance without having the need to do the routing himself, OJP should provide such information. ⚠CR OJP: consider to add this feature (distance instead of a given from/to route) in OJP.
OJP.OJPFareRequest.TripFareRequest.Trip.TripLeg.TimedLeg/ContinuousLeg.Leg.ServiceDeparture.TimetabledTime	departureTime	OK
OJP.OJPFareRequest.TripFareRequest.Trip.TripLeg.TimedLeg/ContinuousLeg.Leg.ServiceArrival.TimetabledTime	arrivalTime	OK
OJP.OJPFareRequest.Params.Traveller	nrOfTravelers	No element for number of travellers in OJP. Would need to be calculated from the number of traveller elements (or set to 1, if not available). Travellers is always set to 1.
OJP.OJPFareRequest.Params.Traveller.Age	travelers.traveler.age	OK. Set to 20, if no Traveller exists.

n/a	travelers.traveler.isValidated	information if the travellers identity and properties have been verified by the MaaS provider. Not needed. ⚠CR OJP: Check if OJP might need this for OJP 2.0. Remark: It was decided that OJP is not for booking and does not convey “private” information, this should not be part of OJP 2.0.
n/a	travelers.traveler.referenceNumber	No referenceNumber of traveller found in OJP Fares. Would only work with a global referenceNumber, where a passenger would have the same number for every operator? Not needed. It was checked whether OJP 2.0 could do this. However, as OJP does not deal with personal information, it was agreed to not add such an element.

OJP.OJPFareRequest.Params.EntitlementProduct: Doc: [a specific form of TRAVEL DOCUMENT in TM and NeTEx] a precondition to access a service or to purchase a FARE PRODUCT issued by an organisation that may not be a PT operator (eg: military card, concessionary card, etc) Is of type NMTOKEN. AND OJP.OJPFareRequest.Params.Traveller.PassengerCategory: Doc: sequence of all passenger categories, for which this FareProduct is valid. Enum Adult, Child, Senior, Youth, Disabled. AND OJP.OJPFareRequest.Params.SalesPackageElementRef: Doc: Id of a FareProduct that the passenger already holds and that may be used for the travel or parts of it. Is of type NMTOKEN.	*travelers.traveler.cardTypes* cardTypeStructure: TOMP-API documentation on cardTypeStructure: *"A generic description of a card, asset class and acceptors is only allowed for DISCOUNT/TRAVEL/OTHER cards. Card types: [ID, DISCOUNT, TRAVEL, BANK, CREDIT, PASSPORT, OTHER]* *type Enum:* *[ID, DISCOUNT, TRAVEL, BANK, CREDIT, PASSPORT, OTHER]* *subType string* *For use in case of OTHER. Can be used in bilateral agreements.* *assetClass* *These classes are taken from the NeTEx standard, but ALL and UNKNOWN are removed. On the other hand OTHER and PARKING are added.* *Enum:* *[AIR, BUS, TROLLEYBUS, TRAM, COACH, RAIL, INTERCITYRAIL, URBANRAIL, METRO, WATER, CABLEWAY, FUNICULAR, TAXI, SELFDRIVE, FOOT, BICYCLE, MOTORCYCLE, CAR, SHUTTLE, OTHER, PARKING, MOPED, STEP]* *acceptors [* *references to accepting parties, only if applicable*"	cardTypes could be used by the operator to define fare products. Of the OJP Elements, EntitlementProduct and PassengerCategory fit this narrative, although they are only defined as NMTOKEN or ENUM values and do not provide the granularity of TOMP-API. Also: EntitlementProduct is currently not defined by traveller, but for the whole request. ➔will need further study.
n/a	travelers.traveler.licenseType: licenseTypeStructure	Describes the licences a traveller has to use specific vehicle types (car, etc.). NO OJP Fares element to forward this data. Not needed for fare information but must be added for the availability requests. ➔not needed here.

n/a	travelers.traveler.requirements requirementsStructure Requirements from the end user side. source string if obsolete, it is referencing the travellers' dictionary (https://github.com/TOMP-WG/TOMP-API/blob/master/docu-ments/CROW%20passen-ger%20characteristics.xlsx) category* string references to the first column of the specification initial values [HR, AV, HV, AB, AER, K, ZR, RR] number* string minLength: 2 maxLength: 2 references to the second column of the specification type string conditionally extra information, ref-erencing to the 3rd column memo string extra field for detailed information, not standardized.	Not needed to get fare infor-mation. ➔not needed.
n/a	travelers.traveler.knownIdentifier: identifier for this traveler in the per-sonal data store. This identifier can be used to get personal infor-mation from the provider specified in the "knownIdentifierProvider"	Not needed for fare information ➔not needed.
n/a	travelers.traveler.knownIdentifi-erProvider: provider for personal information. Can be a URI or iden-tifier.	Not needed for fare information ➔not needed.
OJP.OJPFare-Request.Params.ZonesAlradyPaid	n/a	No concept found in /planning/in-quiries to take into account when a zone has already been paid. ➔not needed.
n/a	useAssets: The specific asset(s) the user wishes to receive leg op-tions for	Not needed for fare information ➔not needed.

n/a	userGroups: Id(s) of user groups that the user belongs to. This provides access to exclusive assets that are hidden to the public. Id's are agreed upon by TO and MP.	Not needed for fare information. ➔not needed.
n/a	useAssetTypes: The specific asset type(s) the user wishes to receive leg options for	Not needed for fare information. ➔not needed.

3.1.1 Example OJPFareRequest

On the following pages, an example of an OJPFareRequest is shown.

The example is rather large for a request, mainly because the whole trip structure must be sent to the service. OJP services are stateless, thus, no prior query results or other state information is cached on the server.

In contrast, TOMP-API services are stateful. Bookings objects are cached on the server, and referring to a booking-id will be enough to bring up the booking object again.

```
<?xml version="1.0" encoding="UTF-8"?>
<OJP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.siri.org.uk/siri"
xmlns:ojp="http://www.vdv.de/ojp" version="1.1-dev" xsi:schemaLoca-
tion="http://www.siri.org.uk/siri ../../../OJP.xsd">
  <OJPRequest>
    <ServiceRequest>
      <RequestTimestamp>2020-01-19T12:00:00Z</RequestTimestamp>
      <RequestorRef>MyClient</RequestorRef>
      <ojp:OJPFareRequest>
        <RequestTimestamp>2020-01-19T12:00:00Z</RequestTimestamp>
        <MessageIdentifier>231231-231</MessageIdentifier>
        <ojp:TripFareRequest>
          <ojp:Trip>
            <ojp:Id>192391231</ojp:Id>
            <ojp:Duration>PT55M</ojp:Duration>
            <ojp:StartTime>2020-01-19T12:00:02Z</ojp:StartTime>
            <ojp:EndTime>2020-01-19T12:57:00Z</ojp:EndTime>
               <ojp:Transfers>1</ojp:Transfers>
            <ojp:Leg>
              <ojp:Id>1231123</ojp:Id>
              <ojp:ContinuousLeg>
                <ojp:LegStart>  <!-- TOMP-API is leg based -> consider the relevant leg -->
                  <ojp:GeoPosition>  <!-- the position is one of the crucial elements -->
                    <Longitude>5.1</Longitude>
                    <Latitude>20.1</Latitude>
                  </ojp:GeoPosition>
                  <ojp:Name>
                    <ojp:Text>Origin</ojp:Text>
                  </ojp:Name>
                </ojp:LegStart>
                <ojp:LegEnd>
                  <ojp:GeoPosition>
                    <Longitude>5.2</Longitude>  <!-- crucial -->
                    <Latitude>20.2</Latitude>
                  </ojp:GeoPosition>
                  <ojp:Name>
                    <ojp:Text>Destination</ojp:Text>
                  </ojp:Name>
                </ojp:LegEnd>
                <ojp:Service>
                  <ojp:ContinuousMode>demandResponsive</ojp:ContinuousMode>
          <ojp:OperatingDayRef>2023-12-24</ojp:OperatingDayRef>  <!-- crucial -->
          <VehicleRef>ch:1:vehicle:31231:12311</VehicleRef> <!-- crucial -->
                <ojp:JourneyRef>ch:1:sjyid:12931231</ojp:JourneyRef> <!-- crucial -->
              <LineRef>ch:1:slnid:1231912</LineRef>
          <DirectionRef>ch:1:direction:H</DirectionRef>
          <ojp:Mode>
            <ojp:PtMode>bus</ojp:PtMode>
            <BusSubmode>demandAndResponseBus</BusSubmode>
          </ojp:Mode>
          <ojp:PublishedServiceName>
            <ojp:Text>Mybuxi</ojp:Text>
          </ojp:PublishedServiceName>
          <ojp:OperatorRefs>
            <ojp:OperatorRef>ch:1:sboid:1023123</ojp:OperatorRef>
          </ojp:OperatorRefs>
          <ojp:BookingArrangements>
            <ojp:BookingArrangement>
              <ojp:BookingUrl>
                <ojp:Label>
                  <ojp:Text>MyBuxi</ojp:Text>
                </ojp:Label>
                <ojp:Url>https://mybuxi.ch/booking</ojp:Url>
              </ojp:BookingUrl>
            </ojp:BookingArrangement>
          </ojp:BookingArrangements>
        </ojp:Service>
        <ojp:Duration>PT15M</ojp:Duration>
      </ojp:ContinuousLeg>
      <ojp:EmissionCO2>
        <ojp:KilogramPerPersonKm>0.1</ojp:KilogramPerPersonKm>
      </ojp:EmissionCO2>
    </ojp:Leg>
    <ojp:Leg>
      <ojp:Id>8182381231</ojp:Id>
      <ojp:TransferLeg>
        <ojp:TransferMode>walk</ojp:TransferMode>
        <ojp:LegStart>
          <ojp:GeoPosition>
            <Longitude>5.2</Longitude>
            <Latitude>20.2</Latitude>
          </ojp:GeoPosition>
          <ojp:Name>
            <ojp:Text>Destination</ojp:Text>
          </ojp:Name>
        </ojp:LegStart>
        <ojp:LegEnd>
          <StopPointRef>ch:1:sloid:3000:7</StopPointRef>
          <ojp:Name>
            <ojp:Text>Bern, Gleis 7</ojp:Text>
          </ojp:Name>
        </ojp:LegEnd>
        <ojp:Duration>PT2M</ojp:Duration>
      </ojp:TransferLeg>
    </ojp:Leg>
    <ojp:Leg>
      <ojp:Id>1231123</ojp:Id>
        <ojp:TimedLeg>
        <ojp:LegBoard>
          <StopPointRef>ch:1:sloid:3000:7</StopPointRef>
          <!-- example where it could be based on a stop -->
          <ojp:StopPointName>
            <ojp:Text>Bern, Gleis 7</ojp:Text>
```

```
                </ojp:StopPointName>
                <ojp:ServiceDeparture>
                  <ojp:TimetabledTime>2020-01-19T13:02:00Z</ojp:TimetabledTime>
                </ojp:ServiceDeparture>
              </ojp:LegBoard>
              <ojp:LegAlight>
                <StopPointRef>ch:1:sloid:7000:33</StopPointRef>
                <ojp:StopPointName>
                  <ojp:Text>Zürich HB, Gleis 33</ojp:Text>
                </ojp:StopPointName>
                <ojp:ServiceArrival>
                  <ojp:TimetabledTime>2020-01-19T13:57:00Z</ojp:TimetabledTime>
                </ojp:ServiceArrival>
              </ojp:LegAlight>
              <ojp:Service>
                <ojp:OperatingDayRef>2023-01-24</ojp:OperatingDayRef>
                <ojp:JourneyRef>ch:1:sjyid:11:182391231</ojp:JourneyRef>
                <LineRef>ch:1:slnid:11:8</LineRef>
                <ojp:Mode>
                  <ojp:PtMode>rail</ojp:PtMode>
                  <RailSubmode>highSpeedRailService</RailSubmode>
                </ojp:Mode>
                <ojp:PublishedServiceName>
                  <ojp:Text>IC8</ojp:Text>
                </ojp:PublishedServiceName>
                 <ojp:OperatorRefs>
                  <ojp:OperatorRef>ch:1:sboid:100011</ojp:OperatorRef>
                </ojp:OperatorRefs>
                    <ojp:DestinationText>
                  <ojp:Text>Romanshorn</ojp:Text>
                </ojp:DestinationText>
              </ojp:Service>
            </ojp:TimedLeg>
          </ojp:Leg>
        </ojp:Trip>
      </ojp:TripFareRequest>
      <ojp:Params>
        <ojp:FareAuthorityFilter>ch:1:NOVA</ojp:FareAuthorityFilter>
        <ojp:PassengerCategory>Adult</ojp:PassengerCategory>
        <ojp:TravelClass>second</ojp:TravelClass>
        <ojp:Traveller>
          <ojp:Age>25</ojp:Age>
        </ojp:Traveller>
      </ojp:Params>
    </ojp:OJPFareRequest>
  </ServiceRequest>
 </OJPRequest>
</OJP>
```

3.1.2 Example of a resulting TOMP Request (for the ContinuousLeg)

Of the given OJP TripFareRequest example, only the information in the ContinuousLeg (a demand-responsive bus service named “Mybuxi”) and a tiny amount of information for the parameters are used.

According to the use case sequence discussed above, the TOMP-API endpoint will be looked up first in the service catalogue. The lookup key will be either the PublishedServiceName (“Mybuxi”) or the OperatorRefs (“ch:1:sboid:1023123”, which is a Swiss Business Organisation ID).

In the subsequent call to the /planning/inquiries endpoint, the following JSON payload would be posted. The same key would be added as a query parameter “addressed-to”.

```
{
  "from": {
    "coordinates": {
      "lng": 5.1,
      "lat": 20.2
    }
  },
  "to": {
    "coordinates": {
      "lng": 5.2,
      "lat": 20.2
    }
  },
  "departureTime": "2020-01-19T12:00:02Z",
  "arrivalTime": "2020-01-19T12:57:02Z",
  "nrOfTravelers": 1,
  "travelers": [
    {
      "age": 25,
    }
  ],
  "extraInfo": {
    "additionalProp1": {}
  }
}
```

In the case of a TimedLeg, the "from" element would also contain the stopPreference and/or stationId.

Additional context can be stored in extraInfo, the given extension point. Any JSON object may be added here, including the trip structure, if desired. However, any data added here is not defined by the TOMP-API specification.

3.2 Mapping /planning/inquiries response to OJPFareDelivery

The following table analyses the mapping from TOMP-API to OJP response.

TOMP-API may respond to the request for a single "leg" with an array of legs. All FOOT legs are ignored and only the bookable legs are sent back in the OJPFareDelivery. The client then has to integrate the legs with new footpaths into the whole trip. This may cause problems and require a re-calculation.

Table 2: Mapping from TOMP-API /planning/inquiries Response to OJPFareDelivery.

TOMP-API /planning/inquiries Response	**OJPFareDelivery**	**Remarks and Actions**
error.errorcode (http)	OJP.FareResponse.FareResult.TripFareResult.ErrorMessage.Code	OK

error.detail (http)	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Er-rorMessage.Text.Text	OK
n/a	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Er-rorMessage.Text.TextId	not found in TOMP-API. ➔ currently not needed.
n/a	OJP.FareRe-sponse.FareRe-sult.TripFareRe-sult.FromTripLegId Doc: Identifies the "valid from" trip leg.	If the goal is to provide a fare over multiple legs, this element should be added in TOMP-API. TOMP-API may return multiple legs. They must be integrated in a correct way into the trip before sending it back. Considered practice is to omit FOOT legs and assume that they are to be recalculated/checked by the client. ➔ currently not needed.
n/a	OJP.FareRe-sponse.FareRe-sult.TripFareResult.ToTri-pLegId Doc: Identifies the "valid to" trip leg.	If the goal is to provide a fare over multiple legs, this element should be added in TOMP-API. TOMP-API may return multiple legs. They must be integrated in a correct way into the trip before sending it back. Considered practice is to omit FOOT legs and assume that they are to be recalculated/checked by the client. ➔ currently not needed.
n/a	OJP.FareResponse.Fare-Result.TripFareResult.Pas-sedZones	Element for passedZones does not exist in TOMP-API. ➔ currently not needed.
options.booking.id	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Fare-Product.FareProductId Typ= NMToken	Even though an id is available in the TOMP-API structure, no id is sent in the /plan-ning/inquiries response usually, and it is dis-couraged in the documentation. If it is pro-vided, it should be put into the BookingId in OJP.OJPFareDelivery.FareResult (in OJP 2.0)

pricing.fare.parts.farePartStructure: Documentation: "*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. With 'MAX' you can specify e.g. a maximum of 15 euro per day. Percentage is mainly added for discounts. The scale properties create the ability to communicate scales (e.g. the first 4 kilometres you've to pay EUR 0.35 per kilometre, the kilometres 4 until 8 EUR 0.50 and above it EUR 0.80 per kilometre).*"		
pricing.fare.parts.farePart.name	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductName	OK Pricing can be set in TOMP-API on the leg and on the booking. It is considered good practice by a TOMP-API service to fill in both.
pricing.fare.parts.farePart.amount type: float	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductPriceGroup.Price type: decimal	OK
pricing.fare.parts.farePart.amountExVat type: float	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductPriceGroup.NetPrice type: decimal	OK
pricing.fare.parts.farePart.currencyCode ISO 4217 currency code	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductPriceGroup.Currency ISO 4217 currency code	OK

pric-ing.fare.parts.farePart.vatRate type: float	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Fare-Product.FareProductPrice-Group.VatRate enum: no, full, half, mixed, unknown	Unclear why VatRate in OJP is an enumera-tion. OJP Enum will be changed in OJP 2.0, will be percentage. ➔ To be ignored at the moment.
options.pricing.fare.class type: string doc: "in the future we'll set up an enumeration of possible "fare classes". For now it's free for-mat."	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Fare-Product.FareProductValidi-tyGroup.TravelClass Is of type NMToken with enumeration [all, first, second, third, busi-ness, economy].	Mapping from string in TOMP-API to an enumeration in OJP might be difficult. If no classes specified, use "all", else for public transport use "first" and "second" TOMP-API: Until the "future" has arrived, define the OJP enumeration as best prac-tice in TOMP-API
options.booking.cus-tomer.cardTypeStructure	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Fare-Product.FareProductValidi-tyGroup.RequiredCard Doc: "*One or more traveller cards that are needed for purchase of this Fare-Product. In most cases trav-eller cards offer discounts, f.e. BahnCard50 of Deutsche Bahn.*"	In TOMP-API, the card type is described for a specific customer. ➔ currently not needed.
n/a	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Fare-Product.FareProductValidi-tyGroup.ValidFor is of type string with enu-meration [Adult, Child, Senior, Youth, Disabled]	No validity element found in TOMP-API, possibly TOMP-API only gives fare infor-mation according to the request parameters in CardTypeStructure. ➔ currently not needed.

n/a	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductValidityGroup.ValidityDuration Doc: "*Maximum duration of FareProduct validity starting with purchase of ticket or begin of journey (ticket validation).*"	Not needed, as no purchase of ticket is done. ➔ currently not needed.
n/a	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductValidityGroup.ValidiityDurationText Doc: "*Textual description of maximum validity duration*"	Not needed. ➔ currently not needed.
n/a	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductValidityGroup.ValidityTariffZones Doc: *"patial validity of FareProduct defined as list of fare zones."*	Tariff zones concept is not handled in TOMP-API ➔ currently not needed.
n/a	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductValidityGroup.ValidityAreaText Doc: "*Textual description of spatial validity*."	➔ currently not needed.
n/a	OJP.FareResponse.FareResult.TripFareResult.FareProduct.FareProductBookingGroup.InfoUrl Doc: *"URL to information for this FareProduct"*	This information may not be of use for the customer. ➔ currently not needed.

n/a	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Fare-Product.FareProductBook-ingGroup.SaleUrld Doc: *"URL to buy the Fare-Product online"*	With planning/inquiries, this is probably not used. However, this might be filled in by the transformer. It is assumed that this URL may directly go to the session and not only to the booking service. However, this depends on the im-plementation. As the provision of an orderid is discouraged, it is not that important. Possibly can be filled in by information ob-tained from the service catalogue.
n/a	OJP.FareRe-sponse.FareRe-sult.TripFareResult.Fare-Product.FareProductBook-ingGroup.BookingArrange-ments	Not needed for fare information. Possibly can be filled in by information ob-tained from the service catalogue. ➔ currently not needed.
n/a	OJP.FareResponse.Fare-Result.TripFareResult.Stati-cInfoUrl	Possibly can be filled in by information ob-tained from the service catalogue. ➔ currently not needed.
pric-ing.fare.parts.farePart.vatCoun-tryCode axLength: 2 minLength: 2 example: NL	n/a	➔ currently not needed.
pricing.fare.parts.farePart.type [FIXED, FLEX, MAX]	n/a	This information might be of interest to the customer and should be provided in OJP ➔ currently not needed.
pricing.fare.parts.farePart.kind [DEFAULT, DISCOUNT, SURGE] Doc: *is this the default price or is this an additional part (dis-count, price surge). In case of a DISCOUNT, the amount must always be negative and in case of SURGE it must be positive. This also means, that when you're working with discounts or surges, you have to deliver 2 fareparts, one for the default price and one for the dis-count/surge. This can be used in combination with as well the fixed price parts as with the flex price parts.*	n/a	Does the handling of price change in fare, due to discount, have to be communicated to the customer? Can't it be delivered as an own product? ➔ currently not needed.

pricing.fare.parts.farePart.unit-Type [KM, SECOND, MINUTE, HOUR, MILE, PERCENTAGE]	n/a	unitType is only needed if type FLEX is used. Information of the costs when farePart type FLEX is used, is useful for the customer to get an idea on the price development. ➔ currently not needed.
pricing.fare.parts.farePart.unit	n/a	TOMP-API unit → The number of unitTypes (amount of km, seconds, etc.). ➔ currently not needed.
pricing.fare.parts.farePart.scale-From Doc: *number($float)* *minimum: 0* *in case of scaling, this is the bottom value (f.x. in the first hour 3 CAD, the scaleFrom should contain 0 and the scale-Type HOUR). When scaleTo is used, but this field is missing, it should be assumed it is a 0.*	n/a	We think that scaling information is not of relevance to the customer. ➔ currently not needed.
pric-ing.fare.parts.farePart.scaleTo Doc: *minimum: 0* *the upper value of the scale (f.x. 3 CAD in the first hour, this field should contain 1, scaleFrom 0 and scaleType HOUR)*	n/a	We think that scaling information is not of relevance to the customer. ➔ currently not needed.
pricing.fare.parts.farePart.scale-Type Doc: string Enum: [KM, MILE, HOUR, MINUTE]	n/a	We think that scaling information is not of relevance to the customer. ➔ currently not needed.
pricing.fare.parts.farePart.mini-mumAmount	n/a	We think that this information is not of rele-vance for the customer to get a fare/price for a specific trip? ➔ currently not needed
pricing.fare.parts.farePart.maxi-mumAmount	n/a	We think that this information is not of rele-vance for the customer to get a fare/price for a specific trip? ➔ currently not needed.
pricing.fare.parts.farePart.meta	n/a	Empty in doc, not needed. ➔ currently not needed.

pricing.fare.estimated True, False	n/a	No element in OJP FareStructure to define if pricing is an estimation or not ⚠CR OJP: Element should be added to the OJPFareDelivery. Remark: Added to OJP 2.0
mainAssetType	n/a	AssetTypes are handled in accessibility request and not in fare. ➔ currently not needed.
extraData	n/a	➔ currently not needed.

3.2.1 Example of a TOMP-API planning/inquires response

```
{
    "validUntil": "2020-01-19T12:00:02Z",
    "options": [
        {
            "id": "1231231112231",
            "legs": [
                {
                    "id": "1231231112231",
                    "from": {
                        "coordinates": {
                   "lng": 5.1,
                    "lat": 20.2
                        }
                    },
                    "to": {
                        "coordinates": {
                            "lat": 5.2,
                            "lng": 20.2
                        },
                    },
                    "departureTime": "2020-01-19T12:00:02Z",
                    "arrivalTime": "2020-01-19T12:57:02Z",
                    "assetType": {
                        "id": "bus1",
                        "assetClass": "BUS",
                        "assetSubClass": "bus"
                    },
                    legSequenceNumber: 1,
                    "pricing": {
                        "estimated": false,
                        "parts": [
                            {
                                "name": "Kilometerbasiert normal",
                                "amount": 17,
                                "amountExVat": 15,
                                "currencyCode": "CHF",
                                "class": "economy"
                            }
                        ]
                    }
                }
            ],
            "state": "NEW"
        }
    ]
}
```

⚠CR TOMP-API: TOMP-API contains a bookingState in the booking object. When calling the /planning/inquiries endpoint, however, none of the existing ENUM values makes sense. We therefore suggest that an additional value in the ENUM is introduced in the TOMP-API standard for this kind of response. A suitable name would be SPECULATIVE or NONE).

Pricing can and should be added also on the level of booking.

⚠Be aware that multiple bookings can be returned. For each booking, a FareResult must be returned.

3.2.2 OJPFareDelivery

The code example on the following page provides an example of the OJPFareDelivery generated from the TOMP-API response.

In some cases, pickup and dropoff locations may be different from the coordinates added. This needs a refinement request with some recalculations. Also, the TOMP-API service may add footpaths in this case. Either the transformer or the OJP system will have to deal with it. Currently, we will remove them.

The content of BookingArrangement cannot be filled in from /planning/inquiries. This information has to come either from the service catalogue or from different TOMP-API requests.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ojp:OJPFareDelivery xmlns:ojp="http://www.vdv.de/ojp" xmlns:siri="http://www.siri.org.uk/siri"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLoca-
tion="http://www.siri.org.uk/siri  ../../../OJP.xsd">
  <siri:ResponseTimestamp>2001-12-17T09:30:47Z</siri:ResponseTimestamp>
  <siri:Status>true</siri:Status>
  <siri:ValidUntil>2001-12-18T09:30:47Z</siri:ValidUntil>
  <ojp:Problem>
    <ojp:Type>OJPGENERIC_OTHER</ojp:Type>
  </ojp:Problem>
  <ojp:FareResponseContext>
    <ojp:Operators>
      <ojp:Operator>
        <siri:OperatorRef>ch:1:sboid:123123</siri:OperatorRef>
        <siri:OperatorName>Mybuxi</siri:OperatorName>
      </ojp:Operator>
      <ojp:Operator>
        <siri:OperatorRef>ch:1:sboid:11</siri:OperatorRef>
        <siri:OperatorName>SBB NOVA</siri:OperatorName>
      </ojp:Operator>
    </ojp:Operators>
  </ojp:FareResponseContext>
  <ojp:FareResult>
    <ojp:Id>1231231112231</ojp:Id>
    <ojp:TripFareResult>
      <ojp:FromLegIdRef>1</ojp:FromLegIdRef>
      <ojp:ToLegIdRef>1</ojp:ToLegIdRef>
      <ojp:FareProduct>
        <ojp:FareProductId>mybuxi10001231</ojp:FareProductId>
        <ojp:FareProductName>Kilometerbasiert normal</ojp:FareProductName>
        <ojp:FareAuthorityRef>ch:1:sboid:123123</ojp:FareAuthorityRef>
        <ojp:FareAuthorityText>Mybuxi</ojp:FareAuthorityText>
        <ojp:NetPrice>15</ojp:NetPrice>
        <ojp:Currency>CHF</ojp:Currency>
        <ojp:TravelClass>economy</ojp:TravelClass>
        <ojp:SaleUrl>
          <ojp:Label>
            <ojp:Text>MyBuxi</ojp:Text>
          </ojp:Label>
          <ojp:Url>https://www.mybuxi.ch/booking</ojp:Url>
        </ojp:SaleUrl>
        <ojp:BookingArrangements>
          <ojp:BookingArrangement>
            <ojp:BuyWhen>
              <ojp:PurchaeMoment>onReservation</ojp:PurchaeMoment>
            </ojp:BuyWhen>
            <ojp:MinimumBookingPeriod>PT10M</ojp:MinimumBookingPeriod>
          </ojp:BookingArrangement>
        </ojp:BookingArrangements>
      </ojp:FareProduct>
    </ojp:TripFareResult>
  </ojp:FareResult>
  <ojp:FareResult>
    <ojp:Id>123123123</ojp:Id>
    <ojp:TripFareResult>
      <ojp:FromLegIdRef>2</ojp:FromLegIdRef>
      <ojp:ToLegIdRef>2</ojp:ToLegIdRef>
      <ojp:FareProduct>
        <ojp:FareProductId>öV</ojp:FareProductId>
        <ojp:FareProductName>ÖV</ojp:FareProductName>
        <ojp:FareAuthorityRef>ch:1:sboid:11</ojp:FareAuthorityRef>
        <ojp:FareAuthorityText>NOVA</ojp:FareAuthorityText>
        <ojp:NetPrice>51</ojp:NetPrice>
        <ojp:Currency>CHF</ojp:Currency>
        <ojp:TravelClass>second</ojp:TravelClass>
        <ojp:SaleUrl>
          <ojp:Label>
            <ojp:Text>NOVA</ojp:Text>
          </ojp:Label>
          <ojp:Url>https://www.sbb.ch/NOVA</ojp:Url>
        </ojp:SaleUrl>
      </ojp:FareProduct>
      <ojp:FareProduct>
        <ojp:FareProductId>öV</ojp:FareProductId>
        <ojp:FareProductName>ÖV</ojp:FareProductName>
        <ojp:FareAuthorityRef>ch:1:sboid:11</ojp:FareAuthorityRef>
        <ojp:FareAuthorityText>NOVA</ojp:FareAuthorityText>
        <ojp:NetPrice>75</ojp:NetPrice>
        <ojp:Currency>CHF</ojp:Currency>
        <ojp:TravelClass>first</ojp:TravelClass>
        <ojp:SaleUrl>
          <ojp:Label>
            <ojp:Text>NOVA</ojp:Text>
          </ojp:Label>
          <ojp:Url>https://www.sbb.ch/NOVA</ojp:Url>
        </ojp:SaleUrl>
      </ojp:FareProduct>
    </ojp:TripFareResult>
  </ojp:FareResult>
</ojp:OJPFareDelivery>
```

3.3 Conclusions

OJPFareRequest can be supported and answered by a service based on a TOMP-API /planning/inquiries endpoint. However, some information will be lost.

To facilitate the mapping service, some changes to OJP and TOMP-API should be implemented. This will limit greatly the amount of logic in the mapping service. We plan to propose these changes as change requests to the respective boards.