# Technical description of the initial MaaS Data Space, for testing purposes with Accept Institute

| | |
|---|---|
| **Date** | **july 17, 2019** |
| **Version** | **V0.9 under construction** |
| **Status** | **Concept for consultation with MaaS Service Providers for connection and testing** |

# Contents

# 1 Introduction

The definition of Mobility as a Service (MaaS) in the Dutch design is:

*MaaS stands for the offer of multimodal, demand-driven mobility services, with customized travel options being offered to customers via a digital platform (e.g. mobile app) with real-time information, including payment and finalization of transactions.*

MaaS has no need to be invented, as is develops itself worldwide. The way MaaS will settle in each country, region and city is highly dependent on the existing public transport system and local context. The Netherlands has a mature public transport system (although heavily subsidized), nationally (trains), regionally (trains, light rail and busses) and locally (metro, trams, busses and taxis). The public transport system aims at both private and business travelers and is dependent on subvention by governments. Specific transport for disabled, scholars, and patients is organized locally by government with high grants. For tourists there are some specific tickets in the big cities.

MaaS is intended to contribute to the following policy objectives:

- Livability and use of space (parked (for the long and short-term) vehicles);
- Sustainability (environmental impact in the broadest sense: modal split and emissions);
- Capacity utilization of transport modes (car, public transport, bicycle, etc.);
- Accessibility (people's travel range);
- Increasing social inclusion (mobility needs of special groups).

Currently, most service providers are aiming at travelers from specific transport operators (NS-App) or for specific target groups (employers). Travel planners with the possibility to buy tickets are common, sometimes with complementary transport modalities included (OV-fiets). MaaS providers aim at changing mobility by thinking at ecological and other social goals. This way of thinking does not aim at optimizing one specific existing transport modality, but at the traveler's attitude, preferences and (dis)abilities. The seven main functions in MaaS are:

1. **Personal aspects and preferences**: The MaaS service should support the user's input of personal settings, preferences and limitations.
2. **Plan**: A multimodal travel planner that enables Customers to plan a trip based on current departure and arrival times and on current departure and occupancy information regarding the travel modalities offered by the Transport Providers, based on a location of origin and/or destination and times.
3. **Booking**: Functionality that allows Customers to book a planned trip with different modalities at once in its entirety or a portion thereof of his own choosing with, or purchase from, the Service Provider.
4. **Travel**: Possibility for Customers to travel with a Ticket offered by the Service Provider.
5. **Support**: Support offered by the Service Provider to Customers in Planning, Booking, Travel, Modifying and Paying for the journey (e.g. by Mobile app or telephone).
6. **Adjustments**: The Service Provider will take care for customers of one or more of the travel modes offered by the Service Provider as a result of a calamity or unforeseen event, if one or more cannot be delivered or must be modified, or if the customer himself (during the trip) wishes to change the travel mode.
7. **Payment**: The option offered by the Service Provider to have the Customers pay for the entire trip using one existing payment facility and/or mobility card/subscription, which serves as a means of payment and/or admission ticket for all parts and modalities of the trip.

The MaaS principles for this ecosystem at a high level are:

1. *Commercial level playing field; barriers or distortions of competition are to be combatted;*
2. *Equal accessibility for operators & providers; transport operators and service providers must have easy technical access to the MaaS-platforms. They will not be excluded from a competition point of view.*
3. *Mobility accessible for all end users; users must be free to access the MaaS service provider of their choice; MaaS service providers must give access to the available mobility resources;*
4. *Standardized interfaces for the traveler;*
5. *Standardization of language; the MaaS Eco space uses one common language;*

6. *Cost based decentral basic data infrastructure; the data infrastructure for MaaS is decentralized, non-hierarchic and cost based. Not one organization will control the entire MaaS Eco space.*

# 2 Architecture

The MaaS Ecosystem in perspective leads to a data driven business architecture, a demand driven platform economy. In a platform economy companies are able to drive economies of scale and scope not just within their organizational boundaries but beyond them. They leverage digital channels to build ecosystems of buyers, partners and suppliers. In fact, they transform their business first into demand and data driven digital platforms and the leverage the platforms to build the surrounding ecosystem. Platform economies are anything but blissful, as there are many examples adverse (social) effects in various industries. Monopolies, bad performance of service, collective gathering of data and low margins and low wages are well known.

### *3.1 From system based to data driven*

That is exactly what MaaS aims for. Multiple platforms in de shared MaaS Data Space together supporting the platform economy in the MaaS Ecosystem. Figure 5 shows the structure of the data driven platform economy. The Shared Data Space is the central ax that brings Maas Service Providers into the platform economy. There is no need for providers in the Maas Ecosystem to connect on a 1:1 basis, because the shared data space connects all persons and parties in a safe way. Each person and operator need only one connection to the data space. The MaaS service provider connects them with a service offering.

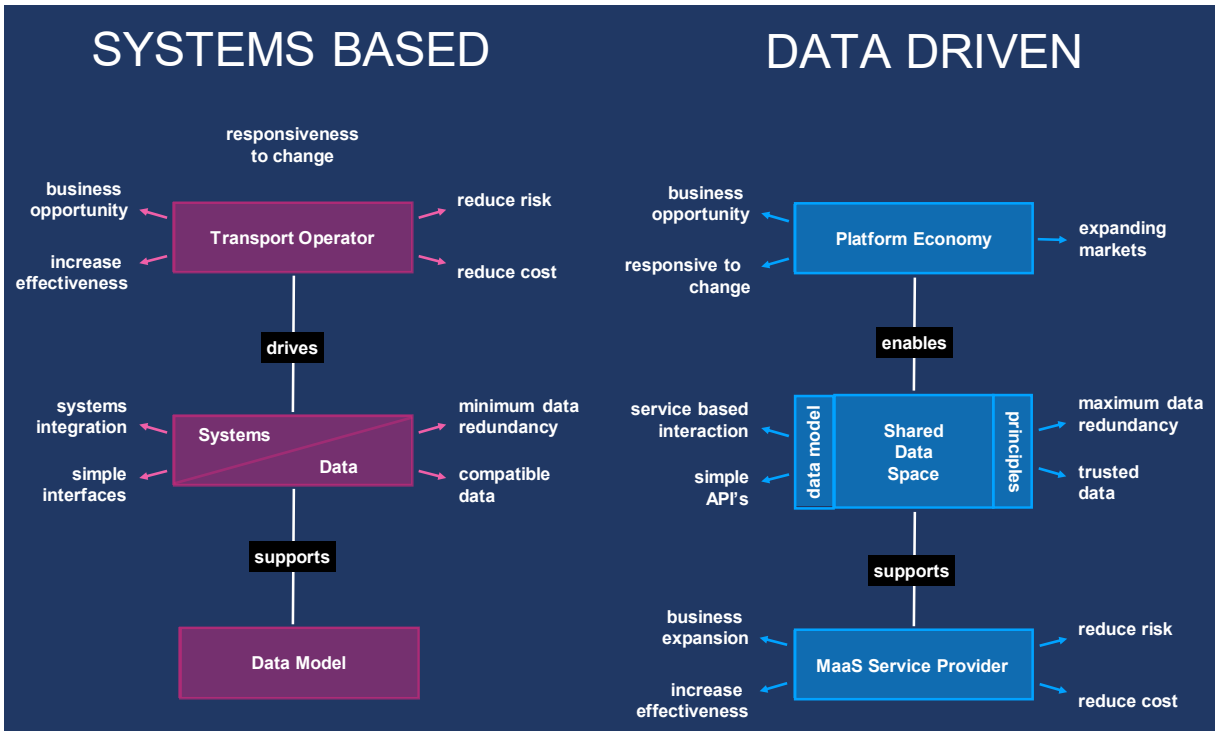



*Figure 1 The new platform economy*

On the left side of the figure the old economy is shown as the opposite. There the operators play the main role and their companies need to be connected through expensive sets of 1:1 interface.

The untenability of this concept is proven by the German industry that decided to develop a new data driven architecture for their business on a platform basis. This architecture, The Industrial Data Space, developed by the Fraunhofer Institute with the industry is used in the design of the MaaS Data Space.

It will take years however for the transformation to take place. While old contracts will be valid for years to go, both worlds must coexist for quite a time.

### *4.1 Key architectural elements in the MaaS Data Space.*

These requirements lead to the following key architectural elements for the MaaS Data Space. The key elements are the building blocks for the MaaS Data Space and should be handled with care when working them out in detail and Mastermind with the results.

1. **Data sovereignty:** It is always the data owner that determines the terms and conditions of use of the data provided (terms and conditions can simply be »attached« to the respective data).

2. **Secure data exchange:** A special security concept featuring various levels of protection ensures that data is exchanged securely across the entire data supply chain (and not just in bilateral data exchange).

3. **Decentral approach** (distributed architecture): The MaaS Data Space is constituted by the total of all end points connected to the Space via the Apps and Data services. This means that there is no central authority in charge of data management or supervision of adherence to data governance principles. In this respect, the MaaS Data Space represents an alternative architecture that is different from both centralized data management concepts (like so-called »data lakes«, for example) and decentralized data networks (which usually have no generally applicable »rules of the game«). What architecture will be used in the end depends on how beneficial each architecture turns out to be in economic terms for each individual application scenario. This is why the MaaS Data Space initiative presumes various coexisting architectures from the outset, converging in the end into one MaaS-architecture.

4. **Data governance** (»rules of the game«): As the Industrial Data Space comes with a distributed architecture, and therefore has no central supervisory authority, data governance principles are commonly developed as »rules of the game«. These rules are derived from the requirements of the users and determine the rights and duties required for data management in to be established in the Ecosystem

5. **Network of platforms and services:** Providers of data can be individual companies, but also »things« (i.e. single entities within the »internet of things«, such as cars, machines, or operating resources) or individuals. Other Data Providers may be data platforms or data marketplaces currently being established.

6. **Economies of scale and networking effects:** The MaaS Data Space provides data services for secure exchange and easy linkage of data. It thereby represents an infrastructure, as using the MaaS Data Space will facilitate the development and use of services (smart services, for example). While these services must rely on data services as offered by the MaaS Data Space, they are not an element of the range of services of the MaaS Data Space themselves. Therefore, economies of scale and networking effects will be critical for the success of the MaaS Data Space: The more participants the MaaS Data Space will have, the more it will become »the place to be« for Data Providers, Data Users, and data service providers alike.

7. **Open approach (neutral and user-driven):** The MaaS Data Space is a user-driven initiative. Regarding the reference architecture model, it is based on a participatory development process, with design decisions being made jointly by the MaaS Service Providers, Transport & Transaction Operators and governmental and transport bodies.

8. **Trust:** (certified participants): It is important for all participants in the MaaS Data Space to trust the identity of each Data Provider and Data User. Therefore all »end points« may connect to the Industrial Data Space via certified software (the »MaaS Data Space Connector«) only. The Connector may also incorporate authentication and authorization functionality.

9. **Coexistence**: The new MaaS dataspace must evolve with respect for current contracts, concessions and other arrangements for Transport and Transaction Processors. Transitional measures to facilitate the transition must be taken in due time.

# 3 Roles in the MaaS Data Space

*Figure 2 MaaS Functionalities*

The MaaS Data Space is role based. The rights of a party in the MaaS Data Space depends on the role(s) it plays. Some roles may be combined, some may rather not be combined, in order to prevent unwanted function mixing. The roles as Industrial Data Space differentiates, are also use in the MaaS Data Space, where they are divided into three categories, business roles, data roles and software roles. All three will be worked out in the following paragraphs. Before going into the roles, the main functionalities of MaaS are summarized.

In the MaaS Eco space the primary roles are differentiated. The main roles are the end user, the MaaS Service Provider, The Transport Operator and the Transaction Processor. Adjoining roles are the Common Data Providers which provide all non-transport data that is needed for planning the jouney and Authorities. Authorities provide (local) data about events, planned maintenance and circumstances, also needed for planning purposes

# 4 The MaaS customer journey

In the Blueprint for an Application Programming Interface (API) from Transport Operator to MaaS Provider V1.0 the functional blocks in the MaaS customer journey are defined. For further details please consult the API document on https://dutchmobilityinnovations.com/spaces/1105/nationale-maas-pilots/articles/news/27615/blauwdruk-api-versie-1-0-voor-maas-ecosysteem

Figure 1 below depicts the main functional and information blocks of the MaaS API. This visualization aims at giving a general overview of the different functional modules within the API. The API is composed of 6 functional blocks. The 2 orange functional blocks (Operator Information & Asset Information) contain fixed information about the transport operator and their assets based on the General Bikeshare Feed Specification (GBFS).

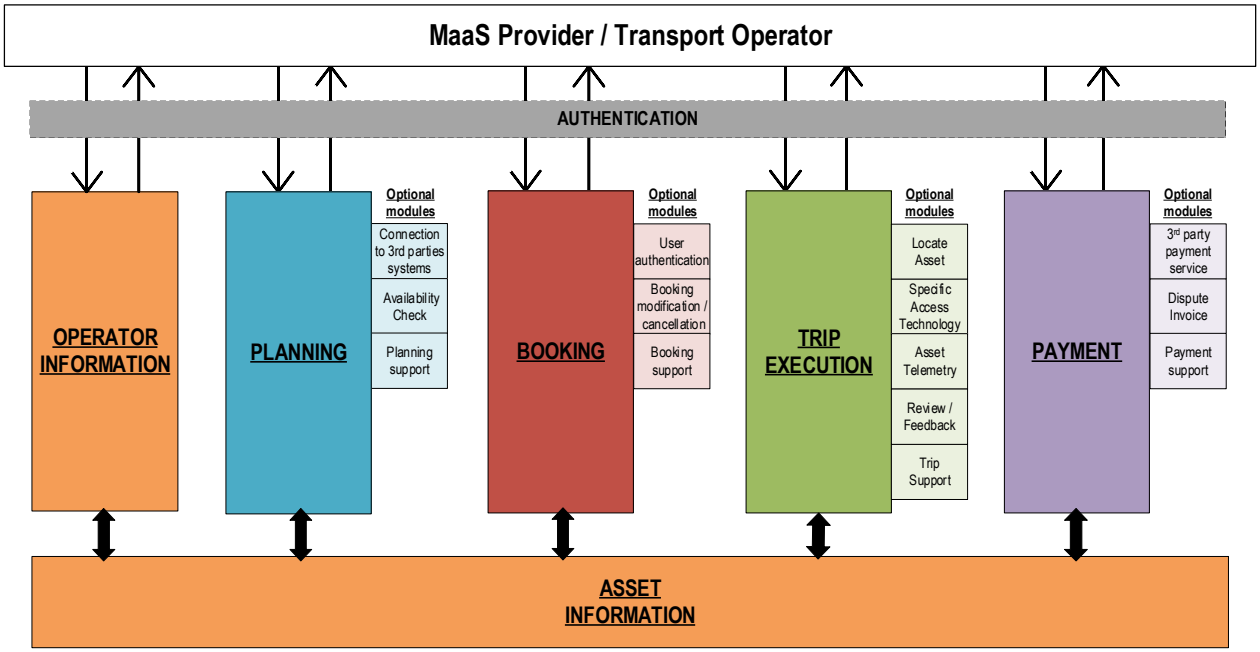



*Fig. 1: MaaS Provider to/from Transport Operator API – Functional Blocks visualization*

The MaaS API functional blocks visualized in Fig. 1 shows the different functions in the interface between MaaS Operator and Transport Operators for the functions of operator information, planning, booking, executing, paying for a trip and providing asset information respectively.

- Operator Information: Gives static information on the operator according to the GBFS(+) standard
- Planning: Gives information about availability, estimated travel time and costs.
- Booking: Allows booking of a specific asset for a specific place, time and date.
- Trip Execution: Allows access to asset and travel during booked period.
- Payment: Allows payment of the service. Supports different business models (i.e. pay-as-you-go or subscription-based).
- Asset Information: Is defined as a separate module that can be used by other modules to supplement API calls with specific asset information where applicable. Assets can be vehicles or for example infrastructural assets
- Optional modules: The more dynamic functional blocks have additional optional modules which are used for execution of sub-processes derived from the main functions which might not be desired or required depending on scope of the MaaS implementation and Business Models.

# 5 The built of the data string


## *5.1 From A to B*

A trip goes from home to work, from work to a client or from theater back home. In MaaS trips are divided in Legs. A leg is defined as moving from one place to another using one specific means of transport in a defined period. The figure shows the core of the data string. Locations are geographic locations. If not available, they will be calculated from for instance the postal code. For privacy reasons, it may be necessary not to use the exact location (I.e. the End User's home) but the center of the postal code.

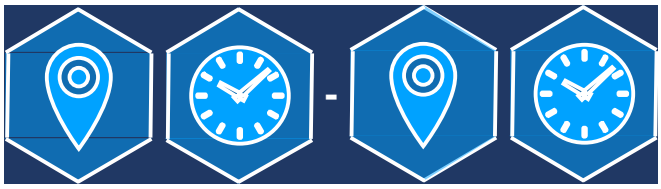


*Figure 3 From A to B*

## *5.2 The Asset*

A leg can only be taken using a specific asset. Every asset within MaaS belongs to a Transport Operator has been granted an ID when licensed for the MaaS Data Space. Every asset within Maas must have its own ID, some by law, others not. The asset data are built as follows:

ASSET ID

COUNTRY ID

TRANSPORT OPERATOR ID → license


*Figure 4 The Asset*

### *5.3 The End User*

The End User, the actual traveler, private or business must also be identified uniquely. The End User can only gain access to MaaS through his Maas Service Provider, who also has a unique identity issued by a license provider. The Maas Service Provider and End User data are built as follows:

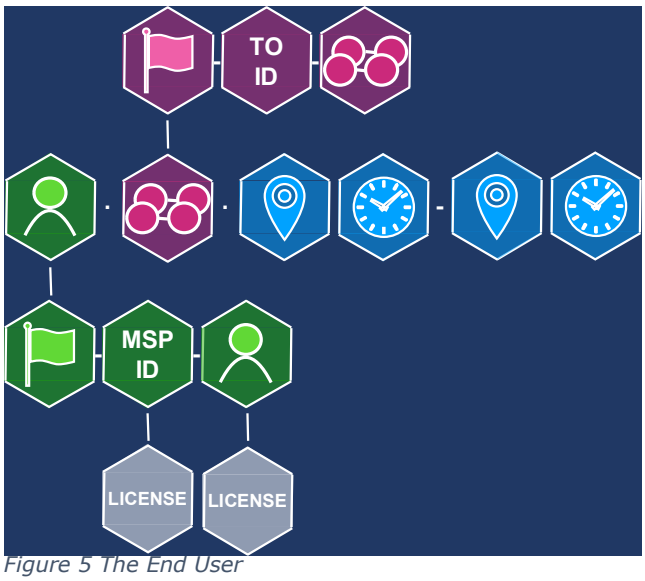



*Figure 5 The End User*

END USER ID

END USER DATA

COUNTRY

MAAS SERVICE PROVIDER ID.

The Maas Service Provider must know the End User to be able to have his itinerary, his preferences and if necessary, his limitations or handicaps. Besides that, his creditworthiness and perhaps driving license(s) must be known. If the End User likes to be assisted on the way, his cellphone number or email address must be available. The identity will preferably be determined only once, including the parameters mentioned above by a validation service.

### *5.4 Type of leg*

The leg is of a certain type, that is described by:


*Figure 6 The Type of Leg*

MODE

*Train*

*Bus*

*Subway*

*Tram*

*Boat*

*Car*

*Bike*

*Moped*

*Scooter*

*Sedgway*

*Watercab*

*On foot*

ENERGY LABEL *A-G (to be decided on*

The various types of mode are defined in the MaaS vocabulary, that describes the language in the MaaS Eco space. The vocabulary is maintained by the MaaS vocabulary provider. In future the type of mode may be expanded, depending on Dutch or EU CO2 policies. The type of mode is decided on by the End User.

### 5.5 Sort of leg

The sort of leg defines the role in which the End User travels, this may vary per leg within a trip. It also describes the sort of travelling service rendered. The sort of leg is described by:

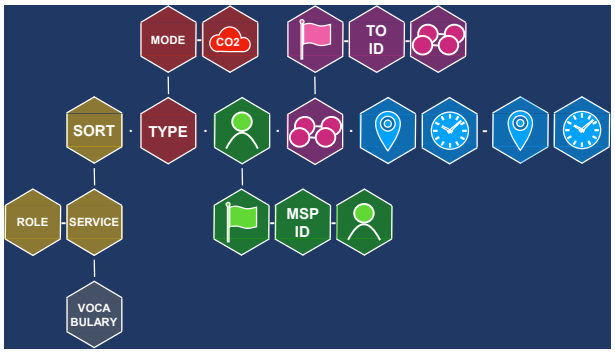



*Figure 7 The Sort of Leg*

END USER ROLE

*Private*

*Commuting*

*Work-work*

*Student (OV)*

*Senior [reduced rate]*

*Juvenile [reduced rate]*

*Disabled [funded transportation]*

SERVICE RENDERED Proprietary transport

Public transport timetabled

Public transport on call

Car pooling / Ride sharing P2P

Ride Hailing / B2C

Subsidized transport on call G2C.

The role and the service to be rendered are often closely connected. A private traveler can travel the way he wants, when not disabled. An employee and a scholar usually have to travel in permitted types of transport.

### 5.6 User Data

User data like role, service, preferred type and similar data result from user defined preferences, (dis)abilities, employer grant, community contract, etc. These private data must be made accessible to the MaaS Service Provider in a practical but safe way, taking into consideration that these data are privacy sensitive.

### 5.7 The full Data string

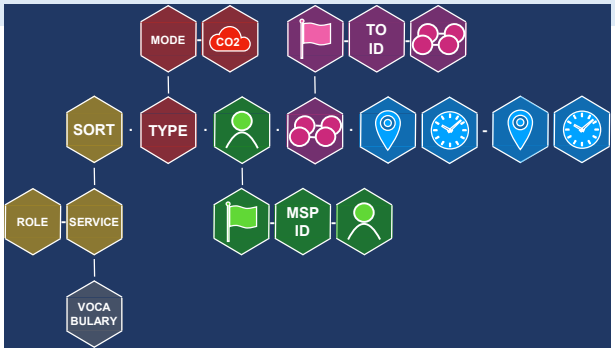
In the previous paragraphs the various parts of the built of the data string were described. When all parts are added together, this leads to this data string:


*Figure 8 The full Data string*

### *5.8 Event type*

One element is not part of the string but belongs to it, the event. The event defines the status of the data string of the leg:

| EVENT TYPE |
| --- |
| Planned leg |
| Booked leg |
| Travelled leg |
| Leg changed by EU |
| Leg changed by TO |
| Leg changed by MSP |
| Leg cancelled by EU |
| Leg cancelled by TO |
| Leg cancelled by MSP |
| Leg cancelled by AUTH |




Herewith the data string SO (booked) and S1 (completed) can always be seen in its context.

### *4.9 MaaS base code tables*

The desciptions of the elements of the datastring above are examples. The complete and actual descriptions are documented in the paper: Maas Base Code Table Vx.x.

# 6 The MaaS Message

MaaS is a data driven Ecosystem, which means there will be a huge number of messages to be sent between the MaaS parties. There are two main types of messages in MaaS. The first type gives information about places where the leg will begin, end or pass through as routes, stop, stations, etc. This type also includes circumstances in which the leg will (likely) take place as weather and traffic jams.

The second type of message is the MaaS Message, in which the data string of the leg is communicated between Maas Service provider and the Operators in the Data Space. This message can be used for retrieving information and giving booking and settlement information from and to Operators. The MaaS Message is built as follows:

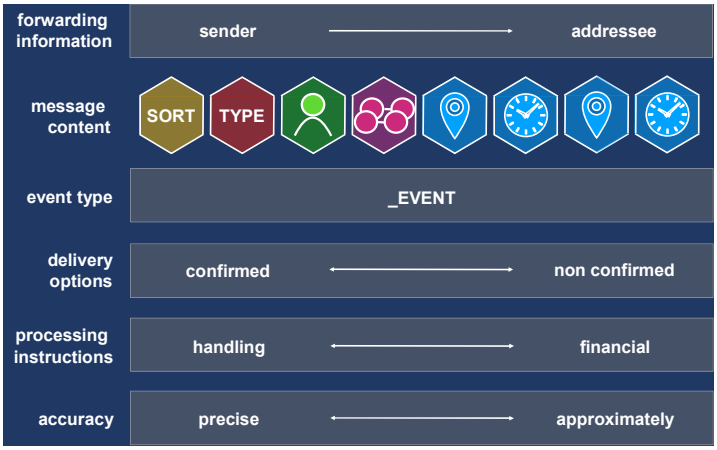



*Figure 9 The MaaS Message*

The Forwarding information is meant for the Message Router, to be able to deliver the MaaS Message at the right address and for the addressee to know who he is dealing with. The Data String is the core of the message and is accompanied by the Event-type, explaining the actual status of the Data String. Delivery of the message may be or may not be confirmed, depending on the preferences of MaaS parties. The processing instructions tell the addressee what he is expected to do with the message. The Accuracy describes how accurate the planning or booking of the leg must be, from precise to approximately in time and place.

These messages will be transferred by various types of API's. complementary information may be passed in by form instance sensors and ticket poles. The figure below shows the variety of ways in which information will be exchanged within the MaaS Data Space.

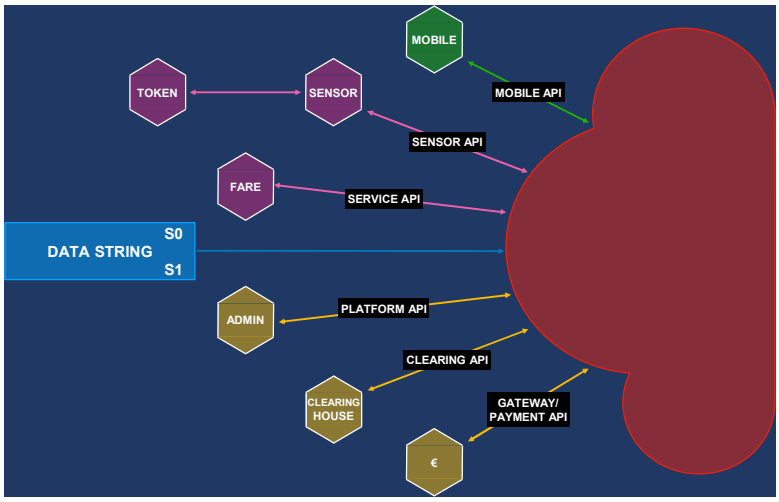



*Figure 10 MaaS API's and connectors*

# 7 Protocol for messaging

Messages within the MaaS Data Space will be transported over the HTTP-protocol.

Various API's will be used (e.g. Mobile-api, Service-api, Platform-api, Sensor-api). These api's are Rest-API's that accept messages over the HTTP-protocol in Json and XML-format (see Figure 10 MaaS API's and connectors).

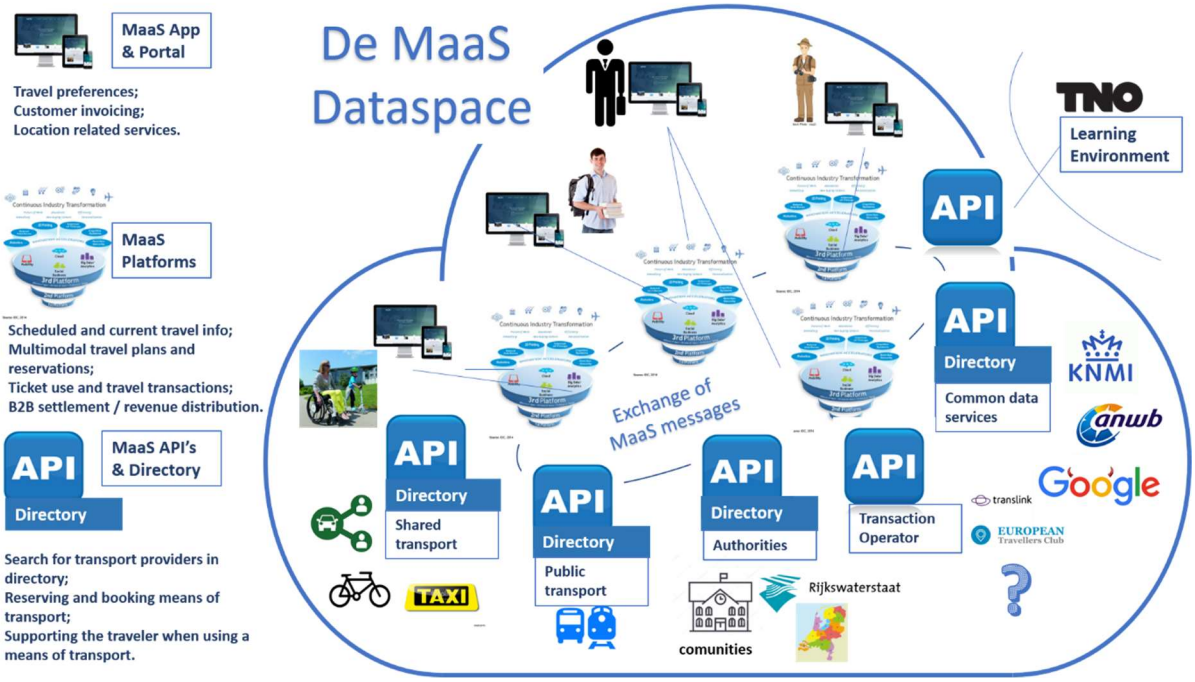
An example of a message to be sent to the Sensor-API is the triggermessage, which is described in chapter 11 of this document.

# 8 API's in the MaaS Dataspace

API's are at the base of the MaaS Dataspace as they connect the parties in the MaaS Dataspace by giving access to and exchanging data. API's are part of a server that generally receives requests and sends responses. Like the mail room in an organization does. API's are connected to services or application that handle the request and prepare the responses.

API's may be divided into different categories:

1. Listener API's. Listener API's are triggered by a sensor and receive one ore more sets of data from the sensor, e.g. in a cardreader in bus or om station;
2. Lookup API's. These API help a service to find specific data on a remote storage, e.g. in a timetable of a connected party;
3. Service API's that receive data from a connected party and trigger a service (inbound) or are triggered by a service to send data (outbound). The recently developed an Application programming Interface (API) from Transport Operator to MaaS Provider is an example of a service API.
4. ;
5. Broadcast API's that sent messages to a group of connected parties, e.g. traffic information and weather updates.



# 9 Specification of the API's for delivering the Data String to the Learning Environment at TNO

### *11.1 Functional specification of the Accept Ecostore*

### *11.2 Technical Specification of the API for the MaaS Service Providers*

Both MaaS Service Providers and Transport Operators will send transactions through the MaaS-Message Routerusing the Sensor-Api (see Figure 10 MaaS API's and connectors) sending a TriggerMessage (see 10.2 TriggerMessage).

The Sensor API is used to connect sensors, e.g. contactless readers (or NFC readers) to the ACCEPT EcoSpace Core (see Figure 11 Accept Ecospace Core). These sensors could also be iris recognition devices or fingerprint scanners.

The Sensor API supports message exchange via a REST interface currently supporting XML & JSON message structures. Due to the message size overhead of XML, JSON format is recommended.

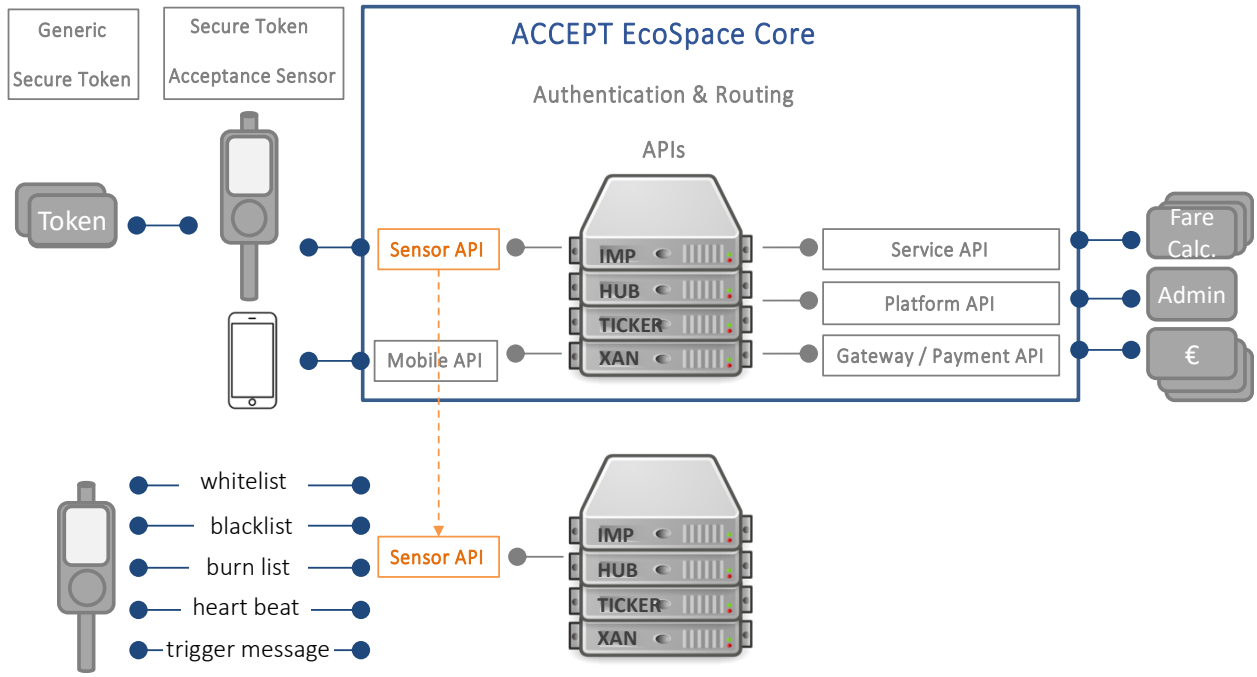



*Figure 11 Accept Ecospace Core*

The detailed specification of the Sensor API is available at ACCEPT Institute. See also: https://maas-nl-hub.westeurope.cloudapp.azure.com/sensor/swagger/ui/index

The link to the swaggerpage of the MaaS-NL-Router is https://maas-nl-hub.westeurope.cloudapp.azure.com/sensor/.

### *9.3 Specification of the message cache for TNO*

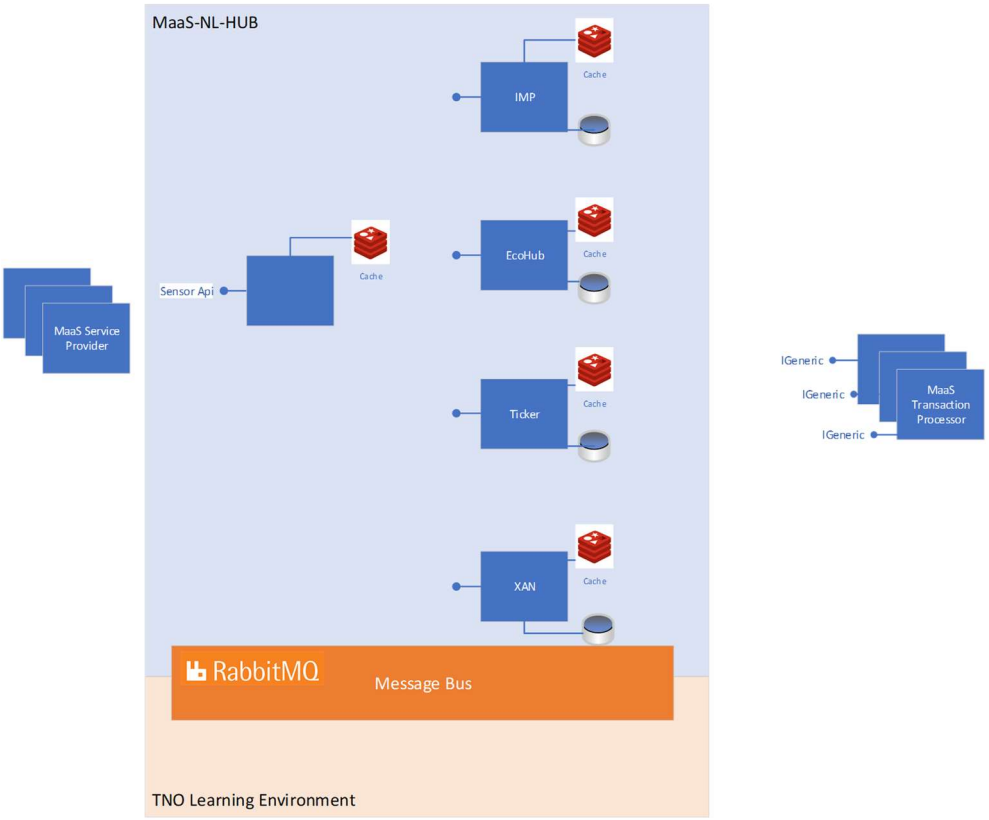
Messages sent through the MaaS-Message Routerare internally sent over to a RabbitMQ message bus. Completed transactions (which are messages that have been processed by the MaaS-Message Router) are anonymized and sent to a queue.

A listener for the Learning Environment Listener(LEL) will be created by TNO based on example code provided by Accept Institute that subscribes to this message. Messages in

this queue will be transformed into data strings by this listener and further processed by TNO.

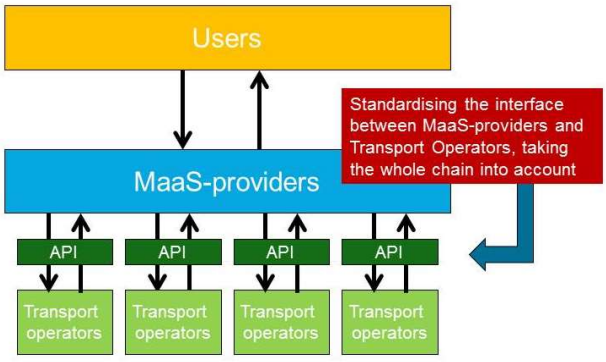
The data strings will have the format as described in Chapter 7

The built of the data string.




# 10 Specification of the API for shared transport (under construction)

## *9.1 Specification of the Shared Bike API*




In this Blueprint for an API from Transport Operators to MaaS Service Providers we look into the necessary functional requirements for the interoperability between transport operators. The goal of this document is to:

- Define the necessary scope for full interoperability between transport operators for the deployment of MaaS services
- Define the necessary parameters and values to fulfill this scope
- Define the available parameters in various already available API's, and propose amendments where applicable.

Version 0.8: https://app.swaggerhub.com/apis-docs/efel85/GTOAS/1.0.2

### *9.2 Specification of the Sensor API*

***Transport Operators will send transactions through the MaaS-Message Routerusing the Sensor-API (see Figure 10 MaaS API's and connectors) sending a TriggerMessage (see ccept Ecostore***

11.2 Technical Specification of the API for the MaaS Service Providers).

The detailed specification of the Sensor API is available at ACCEPT Institute. See also: https://maas-nl-hub.westeurope.cloudapp.azure.com/sensor/swagger/ui/index

### *9.3 Segregation of roles and responsibilities*

The API's give access to the various data sources within the MaaS Dataspace and thus needs safeguarding. In chapter 5, the commercial roles within the dataspace are described. It is not unlikely that consortia within MaaS will combine multiple commercial roles and their access to data. The data roles as described in par. 5.2 can be used to implement restrictions to access to data., creating some segregation of duties. The restriction of access is technically implemented within the API's and the Data Services. The different roles in the supply of software for API's and Data Services must guarantee that the API's and Data Services are installed properly, and work as specified. This also applies to the End User App's that are published in the Appstore's.

### *9.4 Governance over API's in the Ecosystem*

Certification of API's and Data Services and supervision on a certain level must create confidence must create trust among participants of the MaaS Dataspace. This will have to be invested with an independent party to be appointed. At this moment it is not yet possible to foresee the necessary governance for the MaaS Dataspace. For the time being, the program will be in good consultation with the MaaS-parties for implementing a basic governance for the Ecosystem.

### *9.5 Access to API's for MSP and TO in the Ecosystem*

Procedures for connection to and disconnecting of API's and Data Services must be set for the MaaS Data Space. This procedures must technically and legally as easy as possible, but an adequate assessment of the applicant is necessary to ensure confidence in the data space. . At this moment it is not yet possible to foresee the necessary party to perform this function for the MaaS Dataspace. For the time being, the program will be in good consultation with the MaaS-parties for giving access to the Ecosystem.

### *9.6 Quality and certification of API's*

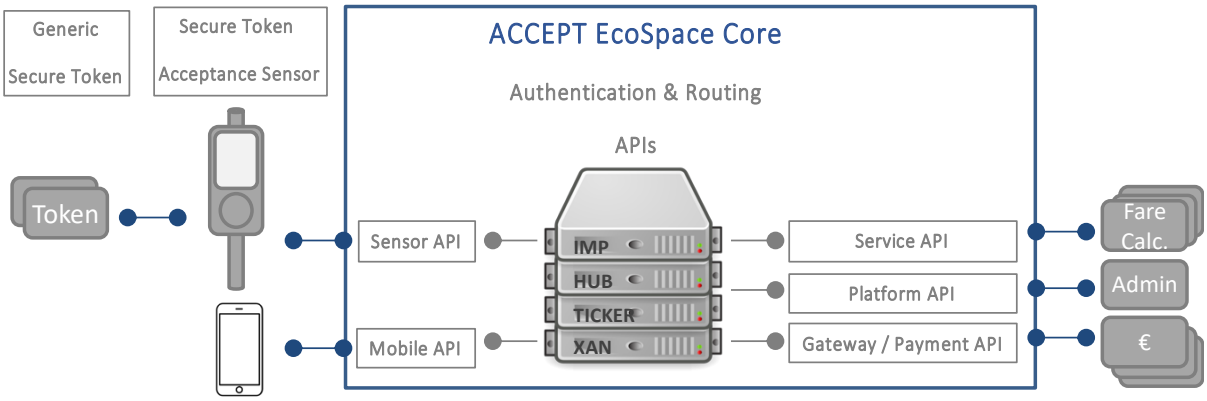
The quality of the different APIs will be tested by Accept Institute when connected to the MaaS Message Routertest environment. For that purpose, a separate test process will be defined according to what is described in chapter 16 of this document.

Before going into production in the pilot environment proper certification of the APIs needs to be in place. It is intended to appoint an independent certification body to certify the APIs.

# 11 The specification of the MaaS Message Router

This chapter provides a high-level description of the ACCEPT EcoSpace Core Software (the MaaS Message Router).

It starts with an overview of the logical components, followed by the overview of the APIs and the micro-services.




### *10.1 The built*

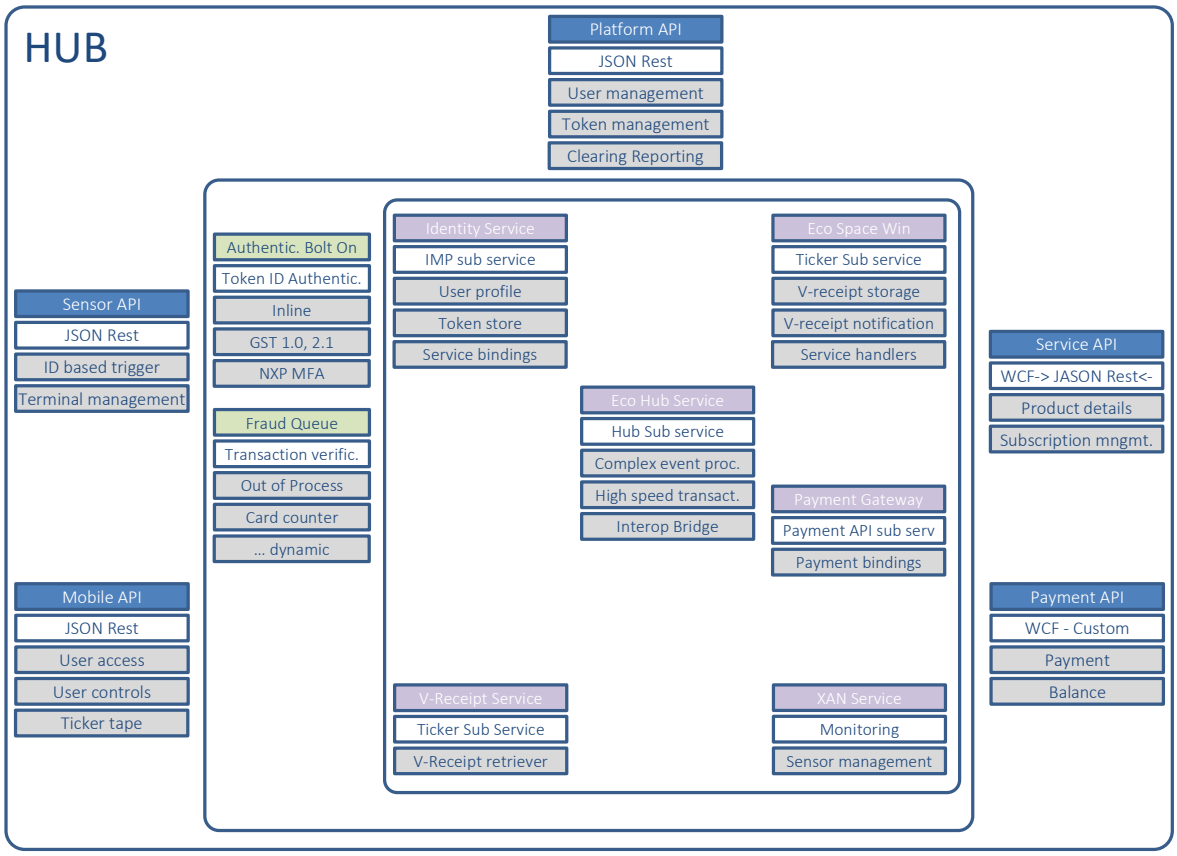
The EcoSpace Core Software consists of the following logical components:

- IMP: Identity Management Platform for Private, flexible and intuitive Account Creation with facilities for binding tokens, services, payment methods to a person's avatar.
- Message Router: High speed transaction authentication and validation, routing and processing router that masks identity and supports complex processing for multi-legged transactions.
- TICKER: Real time transaction overview showing individual consumption of services with Tokens/IDs in the field, a running record of everything tapped.
- XAN: Transaction Acceptance Network, with methods for onboarding tokens, devices, people, services, and payment methods and monitoring their performance and service levels in real time during operation.

Furthermore, the EcoSpace Core Software consists of the following set of APIs for development and integration of external solutions:

- MOBILE API: Account Creation, APP creation, Ticker Views.
- SENSOR API: For device makers to attach their devices to the XAN.
- SERVICE API: For services like Fare Calculation, Park&Ride, ticketing systems and other often travel related multifunctional services such as rentals and in-station lockers.
- GATEWAY / PAYMENT API: High speed secure router for normalizing custom developed bank connections to a universally consumable payment method.
- PLATFORM API: To afford users of the Platform API (owners of a particular Message Router) access to administrative functions related to Account Creation, Token Registration, and Transaction Reporting.

Logical components such as IMP, Message Router, Ticker and XAN, are comprised of underlying micro services.




For a detailed description of the specifications of the Message Routerwe refer to Accept Institute.

### *10.2 TriggerMessage*

Below the most important API-call, i.e. the TriggerMessage, is described. This call enables a user or a system to send a transaction through the MaaS Message Router.

A TriggerMessage is a transaction message that is being sent to the Sensor-API of the MaaS Message Router. A sensor is basically any (registered) device that can register an event, identifying the user or token that initiated the transaction.

A TriggerMessage contains the following blocks of data, which are described below:

- Transaction
- Tokens
- Sensor
- Service
- ServiceRequestData

### *10.3 Transaction*

The transaction-block contains data that should uniquely identify the transaction. In this block the timestamp of the transaction, the counter of the sensor, the sensorId and an externaltransactionId is registered.

The sensorId must be pre-registered at the MaaS Message Router, and for each TriggerMessage the sensorId will be verified against the sensor-registrationlist.

```
        "Transaction": {
                "PropertyBag": null,
                "ReferencedTransaction": null,
                "Timestamp": "20181214110258903+0100",
                "Counter": 636803821789039655,
                "SensorId": "de928fa2-a2c6-46d5-9583-794376bb9802",
                "ExternalTransactionId": "Test-App-636803821789039655"
        }
```

### *10.4 Tokens*

The tokens-block contains a list of (at least one) tokennumber plus tokentype. For certain tokentypes additional tokendata can be added in the propertybag (like tokencounters, TMAC's etc.)

```
        "Tokens": [{
                "TokenValue": "44200000000000000323",
                "TokenType": "GST21",
                "PropertyBag": []
        }]
```

### *10.5 Sensor*

The sensorblock contains two objects, i.e. the identifiers-object and the SensorLocation object.

The identifiers-block can contain additional information about the sensor like a serialnumber or a description or a device-identifier. This data will not be verified, but will be logged.

The SensorLocation is an optional block and can contain latitude and longitude, or cell-info which can be used to estimate a location.

```
        "Sensor": {
                "Identifiers": [{
                        "IdentifierValue": "123456",
                        "IdentifierType": "TEST"
                }],
                "SensorLocation": {
                        "Latitude": 52.3782272,
                        "Longitude": 4.89759,
                        "Altitude": 0.0,
                        "CellId": 0,
                        "LocationAreaCode": 0,
                        "MobileCountryCode": 0,
                        "MobileNetworkCode": 0
                }
```

### *10.6 Service*

The serviceblock only contains a serviceId, which is a number that is preconfigured in the sensor and determines which service will be addressed after the message has been verified by the Message Router.

```
        "Service": {
                "ServiceId": 21
                }
```

### *10.7 ServiceRequestData*

Servicerequestdata contains items that are needed by the addressed service to handle the request correctly. This is why this block contains a propertybag, which is a list that can contain any key(value) and any value(value).

For enabling the MaaS Message Routerto send datastring related data to the backend, a first list of items required in the propertybag is defined below. This list will be expanded later.

***Required propertybag items for datastring***

| Item | Description | Example |
|---|---|---|
| **Starttime** | String, notation in format yyyyMMddHHmmsszzz | 20181214110258904+0100 |
| **Startlocation** | | |
| **Endtime** | String, notation in format yyyyMMddHHmmsszzz | 20181214110258904+0100 |
| **Endlocation** | | |
| **Transportprovider** | Name of the transportprovider | GVB |
| **Event** | Defined eventidentifiers for specific transactions | S1 |


```
"ServiceRequestData": {
        "RequestInternalIpAddress": null,
        "RequestExternalIpAddress": null,
        "RequestSensorLocalTimestamp": "20181214110258904+0100",
        "Amount": 0,
        "CurrencyCode": "EUR",
        "RequestMode": "1",
        "PropertyBag": [{
                "Key": "starttime",
                "Value": "20181214110155+0100"
        },
        {
                "Key": "startlocation",
                "Value": "AmsterdamCentraal"
        },
        {
                "Key": "endtime",
                "Value": "20181214110655+0100"
        },
        {
                "Key": "endlocation",
                "Value": "Rokin"
        },
        {
                "Key": "transportprovider",
                "Value": "GVB"
        },
        {
                "Key": "Event",
                "Value": "S1"
        }]
}
```

### *10.9 Complete TriggerMessage body*

The body of the complete transaction will look like this:

[see next page]


```
{
	"Transaction": {
		"PropertyBag": null,
		"ReferencedTransaction": null,
		"Timestamp": "20181214110258903+0100",
		"Counter": 636803821789039655,
		"SensorId": "de928fa2-a2c6-46d5-9583-794376bb9802",
		"ExternalTransactionId": "Test-App-636803821789039655"
	},
	"Tokens": [{
		"TokenValue": "44200000000000000323",
		"TokenType": "GST21",
		"PropertyBag": []
	}],
	"Sensor": {
		"Identifiers": [{
			"IdentifierValue": "123456",
			"IdentifierType": "TEST"
		}],
		"SensorLocation": {
			"Latitude": 52.3782272,
			"Longitude": 4.89759,
			"Altitude": 0.0,
			"CellId": 0,
			"LocationAreaCode": 0,
			"MobileCountryCode": 0,
			"MobileNetworkCode": 0
		}
	},
	"Service": {
		"ServiceId": 21
	},
	"ServiceRequestData": {
		"RequestInternalIpAddress": null,
		"RequestExternalIpAddress": null,
		"RequestSensorLocalTimestamp":
"20181214110258904+0100",
		"Amount": 0,
		"CurrencyCode": "EUR",
		"RequestMode": "1",
		"PropertyBag": [{
			"Key": "starttime",
			"Value": "20181214110155+0100"
		},
		{
			"Key": "startlocation",
			"Value": "AmsterdamCentraal"
		},
		{
			"Key": "endtime",
			"Value": "20181214110655+0100"
		},
		{
			"Key": "endlocation",
			"Value": "Rokin"
		},
```

# 12 Connecting to API’s and the Message Router’s

The procedure for admission and technical acces to the message router is described in the document:

Connecting a MaaS Service Provider (MSP) to the MaaS-NL-Router  Version: draft, 0.1
Date: 21 March 2019

This document is part of the testing procedure and will be updated regularly.

# 13 Test strategy for Message Router & API's

### *12.1 The approach to testing*

API's and Message Routers are software that each provider or operator may built itself or use available software. The developers of the Message Router's and API's are responsible for the system- and functional testing of the software products they produce.

The first Message Router is built and therefore also tested by Accept Institute. They will be responsible for the functioning of the Message Router and - until other choices are made – the testing, acceptance and connection of new Message Router's

*This paragraph describes the test approach. The objective is to inform all who are involved in the test process about the approach, the activities, including the mutual relations and dependencies. It will need further elaboration in the detailed test plans. These test plans need to be abstracted from this test approach.*

Parties within the MaaS Dataspace connect with each other by means of API's. For each API they want to connect to the Message Router, they must follow the administrative admission procedure with Accept Institute as described in the previous chapter. When being admitted, the MaaS-party has to perform two types of testing with the API: the connection test and the end-to-end test (see for more details 16.2).

The connection test must prove that the API is able from within the IT-environment of the MaaS-party to communicate properly with the Message Router. Accept Institute provides the testing facilities for this test. Problems with firewall- and other settings, must be solved by the MaaS-party. Support from Accept Institute is available when necessary.

After the connection test has been successfully completed, the end-to-end test is performed by sending a message to the TNO-API. Later on this type of test will be extended to other API's.

### *12.2 Scope of testing*

*The following system elements are in scope:*

- *MaaS Service Providers systems*
- *Transport Operator systems and Sensors*
- *Learning Environment systems*
- *Transaction Processor systems*
- *MaaS Message Router*
- *Specific service modules:*
    - *…*

*The following testing phases have been defined:*

- *Component Testing Phase, or Unit Testing Phase, eg.:*
    - *configuration testing;*
    - *performance testing.*
- *Integration Testing, or end-to-end Lab Testing Phase. These tests might be performed on a reference implementation that would mimic the actual implementation;*
- *Pre-pilot Lab Testing Phase: Intended to verify each region's specific configuration using pilot hardware/software. This will be same as Integration Testing Phase but will use Pilot specific components;*
- *Acceptance Testing, or User Acceptance Testing.*

### *12.3 Testing facilities*

Accept Institute has built a test MaaS Message Router with APIs to connect with test environments of parties that want to connect to the MaaS Message Router (eg. the MaaS Service Providers, Learning Environment, Transport Operators).

*The MaaS Message Router primarily performs transaction authentication and routing functions in the system. Specialised sub-systems from MSPs, Transport Operators, Transaction Processors, Learning Environment connect through the interfacing to the Message Router. There is a single implementation of the MaaS Message Routerwith ACCEPT. In order to perform lab testing of the Message Router, before services have been developed, stub implementations of the following service could be used, eg.:*

- *Payment Stubs*
- *MSP Stubs*
- *Services Stubs*

*Initial testing is limited to the reference implementation of the Message Router. Some test scenarios would require the service stubs to be connected to the Message Router(e.g. for end-to-end system tests). In such scenarios, the service stubs themselves are not in scope for testing. When the actual services become available, they will be introduced into the test environment, and therefore included.*

### *12.4 The Data Chain Testing process*

Each MSP, TO, or TP that successfully completed the admission procedure as described in chapter 15 will be connect to the AcceptInstitute and TNO for data chain testing. This testing consists of three parts:

1. Implementing of the API and testing the connection to the Message Router. AcceptInstitute will guide the applicant through the process and will give support if necessary. The AcceptInstitute must confirm the test is completed successfully.
2. Test the connection with the TNO-API by sending initial set of MaaS messages. This test is completed when TNO confirms that the messages are received in good order.
3. The MSP starts sharing Data with TNO by sending streams of T0 and T1 datastrings. TNO supervises the quality of the shared data until the quality is at the desired minimum level. As of that moment the data chain test ends.

# 14 Overview of the Knowledge & Learning Environment

The Knowledge and Learning Environment is a core function of the MaaS Ecosystem, as stated in the Program of Requirements. In this environment the data of all the legs booked and completed in MaaS are stored and analyzed. The KLE is not a part of the MaaS Data Space but connected to it. Figure 17 shows the position of the Knowledge and Learning Environment to MaaS.

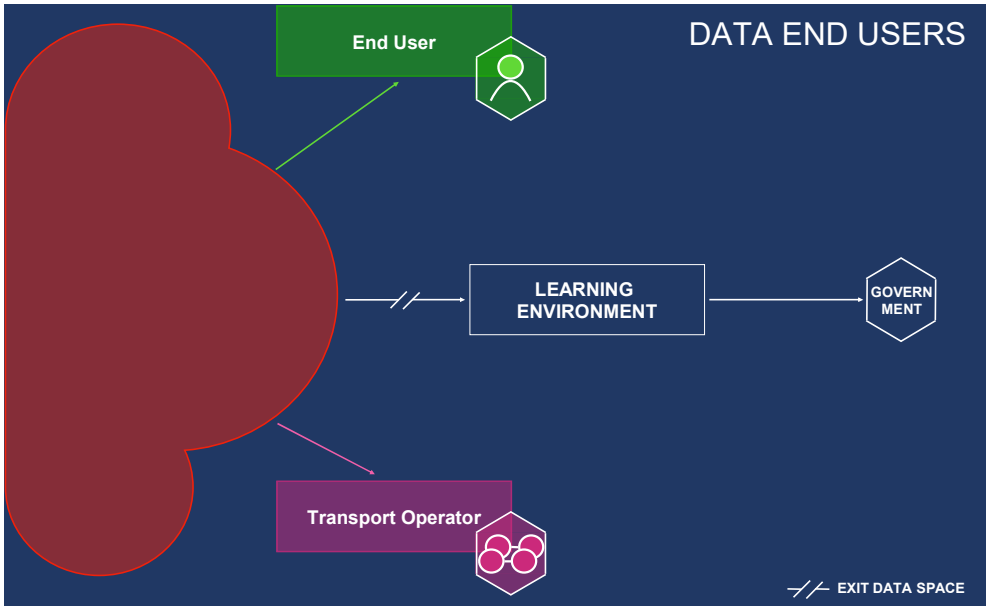



*Figure 12 Users of MaaS Data*

The data strings that are exported to the Knowledge and Learning Environment are complete but anonymized. Each MaaS user has an identifier created for one day only. This ID may be used to combine legs of a user for one day. Connections to legs of other days or other trips cannot and may not be made. Therefore, the data string that is used in the Knowledge and Learning Environment is somewhat manipulated, when leaving the MaaS Data Spaces.

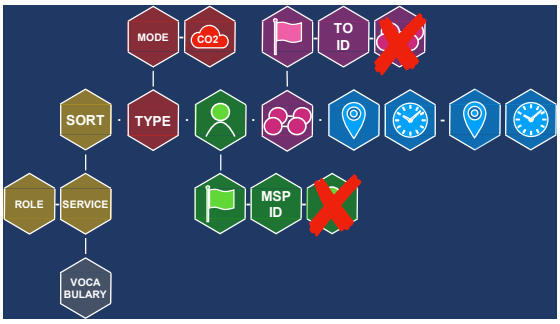



*Figure 13 The Data string in the KLE*

# 15 MaaS Base Code Tables

This chapter describes the base code tables for the functioning of the data string in the MaaS dataspace. The tables are being filled and will be made available by the end of 2019.