



Connecting a
MaaS Service Provider (MSP)
to the
MaaS-NL-Router

Version: final, 1.0

Date: 4 October 2019

Table of Contents

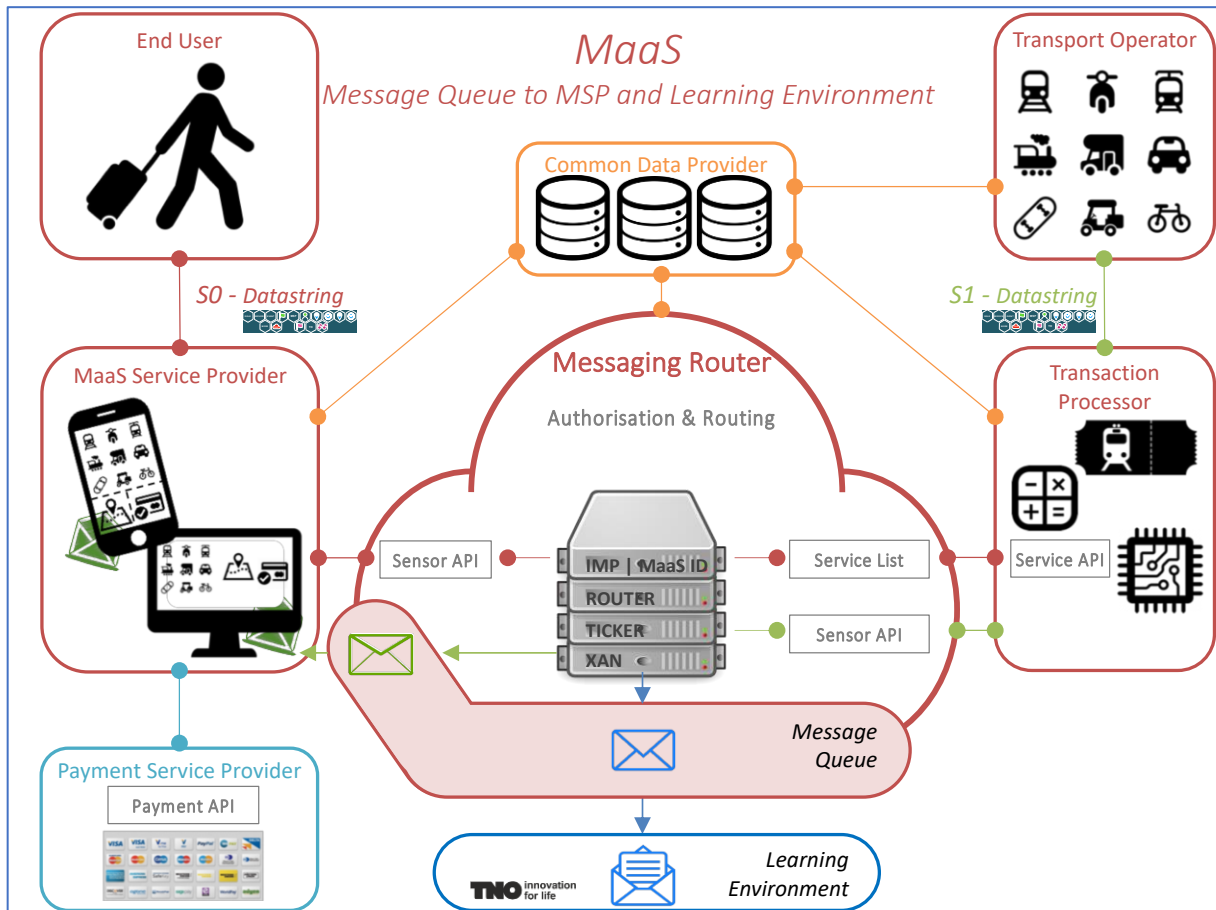
1	Document Status	2
2	Introduction	3
3	Connect an MSP to the MaaS-NL-Router	4
3.1	Registration of a MaaS Service Provider	4
	Creating a certificateset	4
4	Customer onboarding	5
4.1	Connecting a user	5
4.2	Subscribe a user to services	5
5	Send Transactions to the MaaS-NL-Router using TriggerMessage	7
5.1	Transaction	7
5.2	Tokens	7
5.3	Sensor	8
5.4	Service	8
5.5	ServiceRequestData	9
	Required propertybag items for datastring	9
	Complete TriggerMessage body	10
	Appendix A – MSP Application Form	12
	Appendix B – Sensor API	14

1 Document Status

Version	Date	Author	Changes	Changes (by)
0.1	21 March 2019	Vincent Mastenbroek (ACCEPT)	Initial Version	-
1.0	4 October 2019	Vincent Mastenbroek (ACCEPT)	<ul style="list-style-type: none"> - New certificate generation method (makecert.exe is deprecated) - Triggermessage HTTP header elements explained - Extra info concerning Sensor and Service Ids added - Minor textual and grammatical adjustments 	Wouter Priem (ACCEPT)
1.1	10 October 2019	Vincent Mastenbroek (ACCEPT)	<ul style="list-style-type: none"> - Expanded servicerequestdata-propertybag with required and optional fields for datastring 	Vincent Mastenbroek (ACCEPT)

2 Introduction

The MaaS-NL-Router is a transaction-router that validates incoming transactions. Picture below provides the high level architecture of the MaaS.



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

If the validation of the incoming transactions to the MaaS-NL-Router succeeds the transaction is routed to the requested service(s) and afterwards cached in the Message Queue until interested parties have retrieved the relevant data of the transaction.

This document describes how a MaaS Service Provider (MSP) can

- **connect to the MaaS-NL-Router,**
- **onboard its End Users, and**
- **how a transaction can be sent to the MaaS-NL-Router, what parameters are required and optional and how a response is sent back to the caller.**

3 Connect an MSP to the MaaS-NL-Router

3.1 Registration of a MaaS Service Provider

Registering an MSP to the MaaS-NL-Router is an offline / manual procedure. An MSP will have to contact the MaaS-NL-Router administrator from Accept Institute and register details in an application-form (see Appendix A).

When the application is approved, the MSP will be asked to generate a public-key-cryptosystem certificate (RSA-certificate) for signing the messages that will be sent to the MaaS-NL-Router and share the public certificate so the messages can be verified on receipt.

Creating a certificateset

A certificateset consists of two certificates, one private and one public certificate. The public certificate needs to be shared with the MaaS-NL-Router, and the private certificate should be used to sign the messages that will be sent by the MSP to the MaaS-NL-Router.

Below an example is presented of how the certificates can be generated via powershell:

```
New-SelfSignedCertificate -KeyExportPolicy Exportable -Subject "Maas-Service-Provider" -CertStoreLocation "Cert:\CurrentUser\My" -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider"
```

And exported:

```
$cert = (Get-Item -Path cert:\CurrentUser\my\{cert_thumbprint})  
  
Export-Certificate -Cert $cert -FilePath c:\temp\Maas-Service-Provider.cer  
  
Export-pfxCertificate -Cert $cert -Password (Read-Host -AsSecureString -Prompt 'Pfx Password') -FilePath c:\temp\Maas-Service-Provider.pfx
```

Please note that the pfx password will be needed when using this certificate. If necessary, this password can also be left blank.

The MSP will receive a public certificate from the MaaS-NL-Router with which the messages coming from the router should be verified.

4 Customer onboarding

4.1 Connecting a user

When an MSP is connected to the MaaS-NL-Router, end users can be registered by calling the sensor API with the following message:

```
POST <MaaS-NL-Hub-URL>/sensor/v3/Account/Create/{accountId}
```

Where the <MaaS-NL-Hub-URL> is the base-url of the MaaS-NL-Router.

The accountId is a GUID (globally unique identifier) and can be either an empty GUID (00000000-0000-0000-0000-000000000000) or have a valid chosen GUID.

If no GUID is chosen, a new GUID will be generated and registered by the MaaS-NL-Router. Otherwise the chosen GUID will be registered by the router. This makes it possible to register an account from a different MaaS Service Provider for this MaaS Service Provider.

The response message will always contain the registered accountId, whether it was empty in the request or not.

If the registration has been successful, the message will have value 'SUCCESS', otherwise it will be filled with an error message.

ResponseMessage:

```
{
  "data": {
    "accountId": "string"
  },
  "success": true,
  "message": "string"
}
```

4.2 Subscribe a user to services

When a user account is created, first the end user must be subscribed to services before transactions to these services can be sent from the MSP to the Transaction Processor (TP).

The MSP can request the available services by calling the sensor API (see Appendix B) with the following message:

```
GET <MaaS-NL-Hub-URL>/sensor/v3/Sensors/Services
```

Connecting a MaaS Service Provider

This will return a list of Service-objects containing the servicename, the serviceid that has to be referenced and some links to icons that the MSP can display in his app for the end user. The MSP (or the end user using an app of the MSP) can select the services that are needed for the end user, and send a subscription request to the sensor API for each of these services.

The tokentype that has to be used is 'AccountId', the action is 'SUBSCRIBE' or 'UNSUBSCRIBE' and the serviceBinding should contain the (list of) name of the service that is/are requested.

```
PUT <MaaS-NL-Hub-URL>/sensor/v3/Account/SubscriptionChange
```

With body

```
{
  "Tokens": [
    {
      "TokenValue": "294A7695-0626-410D-A82E-C74FDFD0AB4C",
      "TokenType": "accountid",
      "PropertyBag": []
    }
  ],
  "TokenBinding": {
    "TokenType": null,
    "Tokenvalue": null,
    "ServiceBinding": [
      {
        "ServiceName": "MaaSStubService",
        "ParentHubName": "MAAS-NL-HUB"
      }
    ],
    "PaymentMethodBinding": null
  },
  "Action": "SUBSCRIBE"
}
```

5 Send Transactions to the MaaS-NL-Router using TriggerMessage

A TriggerMessage is a transaction message that is being sent to the sensor API of the MaaS-NL-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

Please note that -every- message sent to the sensor API also needs the following 3 elements added to the HTTP header (as described in the MessageSigningSensorAPI pdf found in Appendix B of this document):

- SensorID
- Certificate Thumbprint
- Client-Signature

5.1 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 are registered.

The sensorId is the ID of the registration of the MaaSServiceProvider at the MaaSRouter and once registered can be retrieved from the MaasBaseTables. A valid sensorId must be provided in the triggermessage. For each TriggerMessage the sensorId will be verified against the sensor-registrationlist. The sensorId must be the identifier the MaaS Service Provider received from the MaaS-NL-Router after registering.

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

5.2 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.) The MSP will use the AccountID as tokentype, and the MaaS AccountID of the end user as TokenValue.

```
"Tokens": [{
  "TokenValue": "294A7695-0626-410D-A82E-C74FDFD0AB4C",
  "TokenType": "AccountId",
  "PropertyBag": []
}]
```

5.3 Sensor

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

The identifiers-object can contain additional information about the sensor like a serial number, 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
  }
}
```

5.4 Service

The service-block only contains a service-Id, which is a number of the service / Transaction Processor to be called and determines which service will be addressed after the message has been verified by the router. Available serviceId's can be retrieved using the sensor API of the MaaS-Router (during the pilot phase the ServiceId used will always be '21').

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

5.5 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).

To enable the router to send datastring (see MaaS Data Space, the Dutch Design Document¹) related data to the backend, a first iteration of the list of items required in the propertybag is defined below. This list will be expanded later.

Required propertybag items for datastring

Item	Required	Description	Example
Starttime	Yes	String, notation in format yyyyMMddHHmmsszzz	20181214110258904+0100
Startlocation	Yes		TBD
StartLatitude	No	Double, DD.dddddd° notation	
StartLongitude	No	Double, DD.dddddd° notation	
StartAltitude	No	Altitude in meters	
Endtime	Yes	String, notation in format yyyyMMddHHmmsszzz	20181214110258904+0100
Endlocation	Yes		TBD
EndLatitude	No	Double, DD.dddddd° notation	
EndLongitude	No	Double, DD.dddddd° notation	
EndAltitude	No	Altitude in meters	-3.35
Transportoperator	Yes	id of the transportoperator	1
Event	Yes	Defined eventidentifiers for specific transactions	"S1" or "S0"
Mode	Yes	Specifies the transportmode	500
Energylabel	No	Specifies the energyclass of the transportmode	1
EndUserRole	No	Describes the role of the traveller	200
ServiceRendered	No	Specifies the kind of service rendered	200
Asset	Yes	Specifies the asset used during transport	TBD

The fields Transportoperator, Mode, Energylabel, EndUserRole and ServiceRendered are the identifiers used in the MaaS Base Code Tables, whose api can be found at

<https://maas-nl-hub.westeurope.cloudapp.azure.com/MBT/>

and its swaggerpage is available at

<https://maas-nl-hub.westeurope.cloudapp.azure.com/MBT/swagger/ui/index>

¹ MaaS Data Space, the Dutch Design Document, v0.6, 4 February 2019 (Ministerie van Infrastructuur en Waterstaat).

```
"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": "startlatitude",
    "Value": "52.377727"
  },
  {
    "Key": "startlongitude",
    "Value": "4.8991186"
  },
  {
    "Key": "endtime",
    "Value": "20181214110655+0100"
  },
  {
    "Key": "endlocation",
    "Value": "Rokin"
  },
  {
    "Key": "endlatitude",
    "Value": "52.3701072"
  },
  {
    "Key": "endlongitude",
    "Value": "4.8906203"
  },
  {
    "Key": "transportoperator",
    "Value": "2"
  },
  {
    "Key": "Event",
    "Value": "S1"
  }
  ]
}
```

Complete TriggerMessage body

The body of the complete transaction will look like this:

```

{
  "Transaction": {
    "PropertyBag": null,
    "ReferencedTransaction": null,
    "Timestamp": "20181214110258903+0100",
    "Counter": 636803821789039655,
    "SensorId": "de928fa2-a2c6-46d5-9583-794376bb9802",
    "ExternalTransactionId": "Test-App-636803821789039655"
  },
  "Tokens": [{
    "TokenValue": "294A7695-0626-410D-A82E-C74FDFD0AB4C",
    "TokenType": "AccountID",
    "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": "startlatitude",
      "Value": "52.377727"
    },
    {
      "Key": "startlongitude",
      "Value": "4.8991186"
    },
    {
      "Key": "endtime",
      "Value": "20181214110655+0100"
    }
  ],
  {

```

Connecting a MaaS Service Provider

```

    "Key": "endlocation",
    "Value": "Rokin"
  },
  {
    "Key": "endlatitude",
    "Value": "52.3701072"
  },
  {
    "Key": "endlongitude",
    "Value": "4.8906203"
  },
  {
    "Key": "transportoperator",
    "Value": "2"
  },
  {
    "Key": "Event",
    "Value": "S1"
  }
}]
}

```

Appendix A – MSP Application Form

Name of the Company (MSP)	
Address	
Contact Person	
Contact Details	
Email	
Phone number	

This form can be sent to maasadministrator@acceptinstitute.eu

Appendix B – Sensor API



Sensor API -
FINAL.pdf



2-MessageSigningS
ensorAPIV3-110517-



8-TriggerRequestV3
-110517-1101.pdf



10-RetrieveServiceLi
stV3-210319-1252.p



11-SensorCreateAcc
ountV3-210319-125:



12-SensorSubscripti
onChangeV3-21031: