Deliverable number	Deliverable title	WP number	Lead Beneficiary	Type	Dissemination level	Due date (in months)
D 6.1	Hub Design	WP6	1- Stichting Open Ticketing	Report	Public	3

Hub Design for ETC Platform

Author: **Kley Reynolds**

Checked by: **Janine Zwiers**

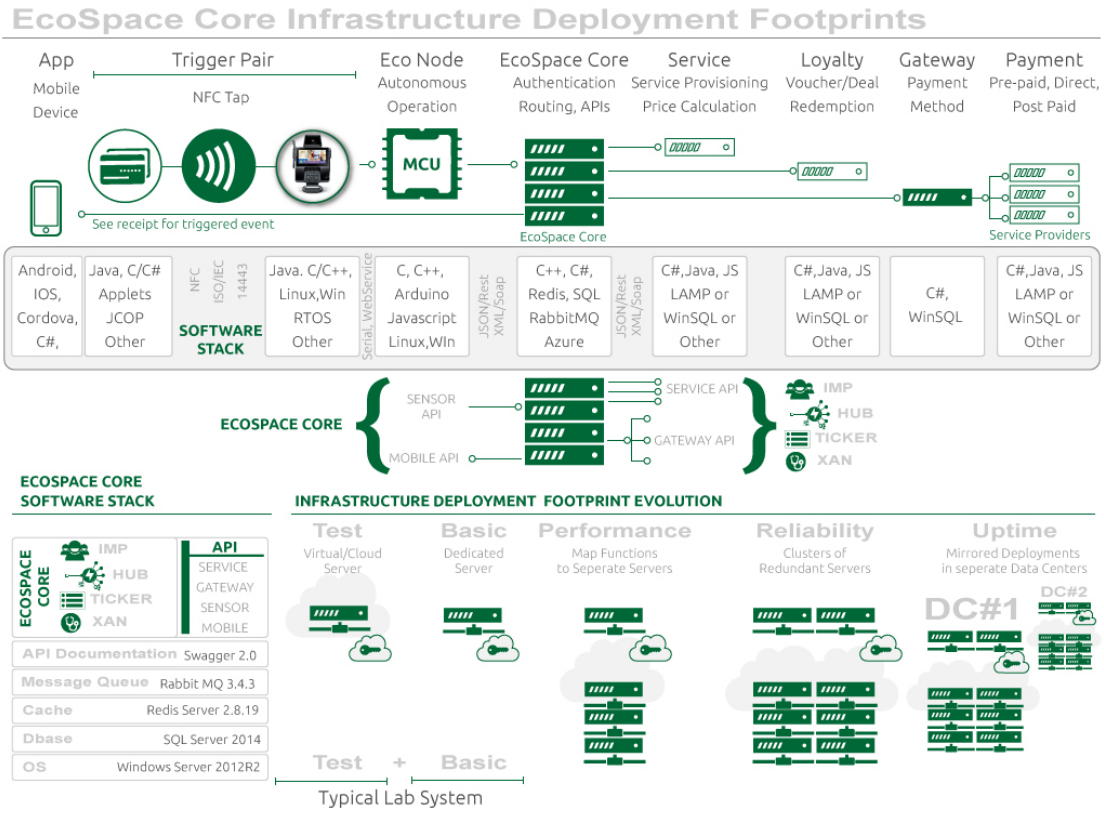
Submitted by: **Kley Reynolds**

Submitted on: **31 – July- 2015**

1. Table of contents

2. Hub Design

This document describes the Hub design of the EcoSpace Core which will be used in the ETC Platform.



The Hub consists of the following 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.
- HUB: High speed transaction authentication and validation, routing and processing hub 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.

And 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 scanning devices to the XAN
- SERVICE API: For services like Fare Calculation, Ticketing systems and other often travel related multifunctional services such as P+R, In-station Lockers and Rentals.

- GATEWAY API: High speed secure router for normalizing custom developed bank connections to a universally consumable payment method.

2.1. Hub Design Audience

This document is intended for people interested in understanding more about the Ecospace (hereto after referred to as HUB) Design. In particular, the concepts behind the platform design and the key components.

The Hub is first and foremost a user centric integration platform, a breadboard for creating connected solutions that can be triggered by humans. In practice it works as follows; the Hub allows humans to create private and secure user accounts, register their Trigger IDs and payment methods and then subscribe and consume connected solutions. These Trigger IDs are used to trigger connected solutions that are built up out of the following connected components:

- Internet of Things which refers to any physical device that can be connected via internet to the Hub.
- Internet of Services which refers to any online service that can be connected via internet to the Hub.

The Hub is designed from the ground up to embrace security by design and privacy by design principles. Despite the concomitant overhead the Hub is capable of completing typical multi legged transactions, from receipt of a trigger message, orchestration of a series of associated transactions ending with a response to the triggered sensor within a few hundred milliseconds. With the majority (~80%) of the latency directly attributable to underlying services, payments services and transmission time. These sub second response times are key to the operation of an EcoSpace and user experience

The Hub is a platform for continuous innovation, advanced user testing and reliable pro active hosting of end solutions. Parties wishing to create a connected solution can focus on the experience and engendering features in a typical solution, e.g. Identity Mmgmnt, high speed complex event processing for high speed processing of complex transactions.

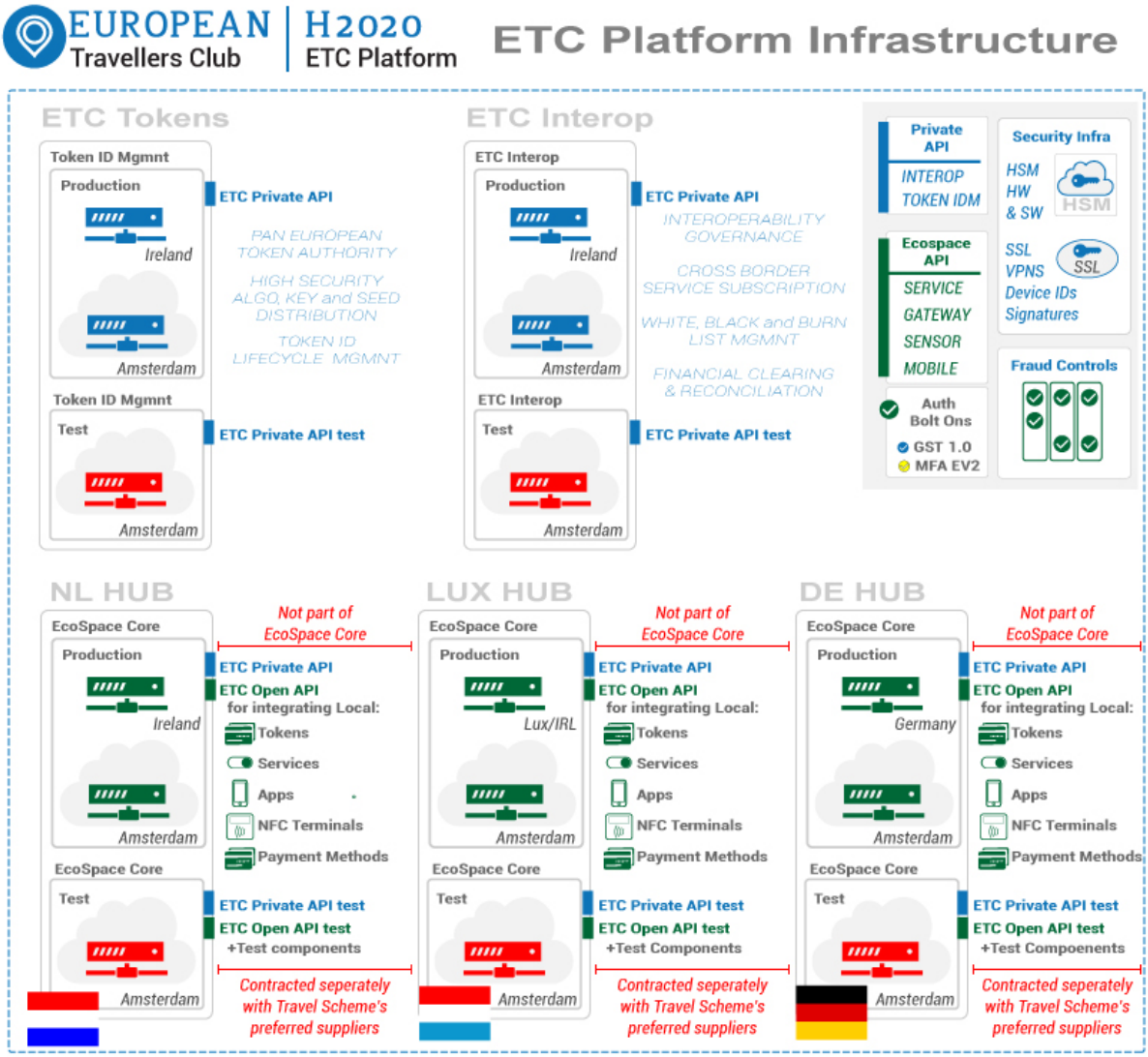
2.2. Living Document

The design as portrayed in this document is subject to revision and will be kept up to date through out the project as we approach the Pilot launch phase.

Expected revisions will flow from:

1) Iterative improvements based upon load tests and other testbench results.
2) Demands met during integration of Travel Scheme's proprietary back end services to meet latency and interoperability requirements defined by the ETC consortium.

3. Hub Design in relation to the ETC Platform

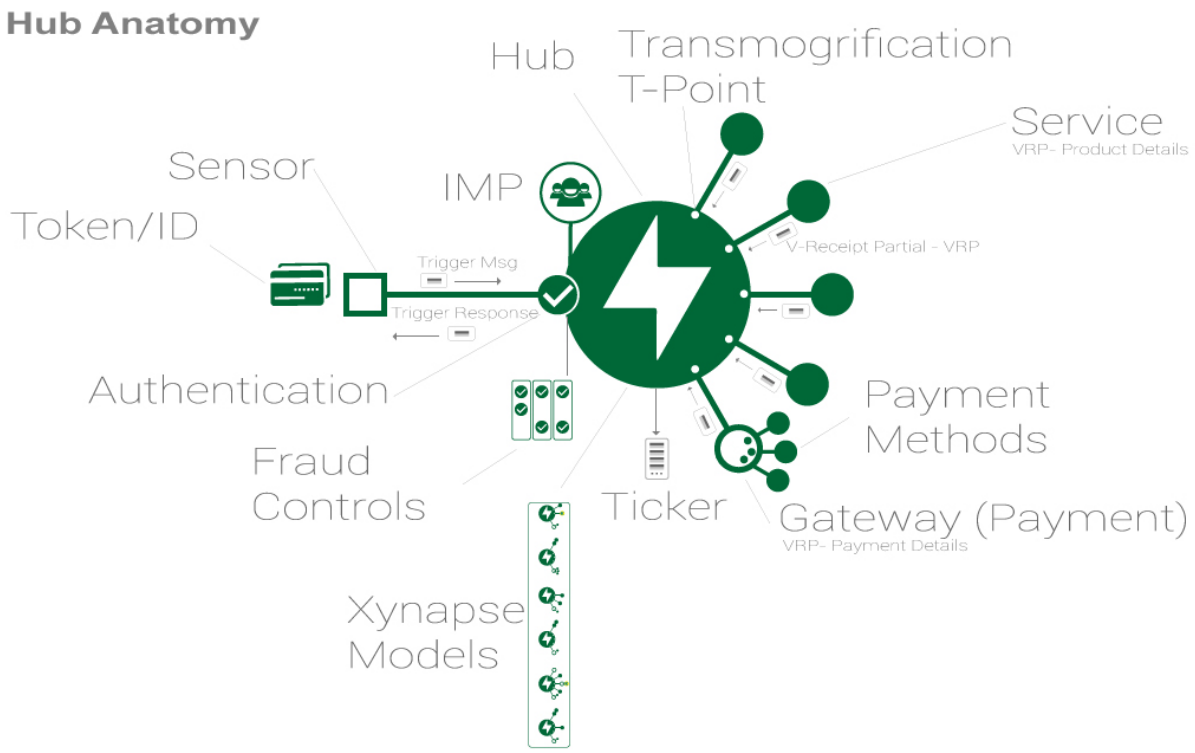


The ETC Platform, consists of a Token ID Mgmnt Module to distribute unique side tokens to Legacy Travel Cards, an Interoperability Hub to allow Travel Scheme Travellers to travel in between participating Travel Schemes based upon Inter-Scheme travel agreements and a series of Travel Scheme Hubs to allow each Travel Scheme to attach their preferred Cloud Ticketing and Cloud Payment services locally.

The ETC Platform will be brought up in a test configuration (see red server in diagram) to aid the participating Travel Schemes in their iterative technical integration activities. Preceding the Pilot launch, a Production Environment for the ETC Platform (green servers) will be brought on-line for real world latency testing and ultimate promotion of the Pilot Components developed in the ETC Test Platform to the production environment.

The constant presence of an ETC Test Platform, makes it a perfect conduit for displaying the latest developments along the way of the various Pilot Use cases in the Travel Lab.

4. Hub Design Anatomy



Each individual hub, of the planned three Hubs for the ETC Platform, can operate standalone to provision ID Based services to consumers within a given Travel Scheme.

They can also be interconnected via an Interop Hub to support interoperable ID based service consumption between hubs.

The Internal working of the Hub is designed to meet exceptional performance targets for latency of transaction completion.

The way one integrates their services with a Hub is via APIs. These API are derived from research into transport and payment method related system messaging and described in detail in the following chapters.

Please keep in mind that these are subject to change as the Pilot integration gets underway. A current version will be posted on the ETC Website for it's ETC Members wishing to integrate their system components.

5. API

5.1. Sensor API

The sensor API is used to connect the EcoSpace Hub to third party sensors, like NFC readers, Iris recognition device, fingerprint etc.
The sensor API supports multiple calls like retrieve whitelist, blacklist, burnlist, send heartbeat and sending a trigger message. Message exchange is done via a REST interface currently supporting XML & JSON message structure. Because of the message size of XML, JSON format is preferred.

Retrieve whitelist

Request messagetype: SensorListRequest

Field	Type	Description	M	SO
ServiceId	long		Y	1
SensorId	Guid		Y	2
Signature	Byte[]	See chapter 0	N	-
SignatureThumbprint	Byte[]	Identification of the used hub-certificate for signing	N	-

Return messagetype: SensorListResult

Field	Type	Description
Collection	String[]	

Retrieve blacklist

Request messagetype: SensorListRequest

Field	Type	Description	M	SO
ServiceId	long		Y	1
SensorId	Guid		Y	2
Signature	Byte []	See chapter 0	N	-
SignatureThumbprint	Byte[]	Identification of the used hub-certificate for signing	N	-

Return messagetype: SensorListResult

Field	Type	Description
Collection	String[]	

Retrieve burnlist

Request messagetype: SensorListRequest

Field	Type	Description	M	SO
SensorId	Guid		Y	1
Signature	Byte array	See chapter 0	N	-
SignatureThumbprint	Byte[]	Identification of the used hub-certificate for signing	N	-

Return messagetype: SensorListResult

Field	Type	Description
Collection	String[]	

Send heartbeat

Request messagetype: HeartbeatRequestMessage

Field	Type	Description	M	SO
SensorLocation	GeoLocation		N	1
SensorConfigurationId	Long		Y	2
PreviousHeartbeatRTT	Long	Previous heartbeat round trip time	N	3
SensorId	Guid		Y	4
Signature	Byte []		N	-
SignatureThumbprint	Byte[]	Identification of the used hub-certificate for signing	N	-

Messagetype Geolocation

Field	Type	Description	M	SO
Latitude	double	DD.dddddd° notation	N	1
Longitude	double	DD.dddddd° notation	N	2
Altitude	double	plus or minus sea level (in meters)	N	3
CellId	int	A GSM Cell ID (CID) is a generally unique number used to identify each Base transceiver station (BTS) or sector of a BTS within a Location area code (LAC) if not within a GSM network.	N	4
LocationAreaCode	int	Location area code (LAC) which is a 16 bit number thereby allowing 65536 location areas within one GSM PLMN.	N	5
MobileCountryCode	int	The Mobile Country Code (MCC) is a three-digit number that used in combination with a Mobile Network Code (MNC) to identify a mobile network operator uniquely.	N	6
MobileNetworkCode	Int	The Mobile Network Code (MNC) is a two digit code (North America) or three digit code (European Standard) that is used in combination with a Mobile Country Code (MCC) to identify a mobile network operator uniquely.	N	7

Return messagetype: HeartBeatResponseMessage

Field	Type	Description
ConfigurationUpdate	SensorConfiguration	If no configurationupdate is available, the value is null

Messagetype: SensorConfiguration

Field	Type	Description
SensorConfigurationId	Long	
InsertDateTime	DateTime	Date & time of sensorconfiguration

		change.
PropertyBag	Dictionary of strings	Container with keyvaluepairs containing the configurationblock for the particular sensor.

Send triggerrequestmessage

When the sensor senses an token, and the token validates successfully locally on the device the sensor sends a triggerrequestmessage to the API.

Request messagetype: TriggerRequestMessage

Field	Type	Description	M	SO
Transaction	Transaction		Y	1
Tokens	Array of BaseTokens		Y	2
Sensor	BaseSensor		Y	3
Service	BaseService		Y	4
ServiceRequestData	ServiceRequestData		Y	5
Signature	Byte []		N	-
SignatureThumbprint	Byte[]	Identification of the used hub-certificate for signing	N	-

Messagetype: Transaction

Field	Type	Description	M	SO
TransactionId	string	TransactionId based on localtimestamp, format: yyyyMMddHHmmssfff	Y	1
Counter	long	Counter of the sensor	Y	2
SensorId	Guid		Y	3
ExternalTransactionId	string	Sensor reference number	N	4

Messagetype: BaseSensor

Field	Type	Description	M	SO
Identifiers	Array of SensorIdentifier		Y	1
SensorLocation	GeoLocation	See chapter **Fout! Verwijzingsbron niet gevonden.**	N	2

Messagetype: SensorIdentifier

Field	Type	Description	M	SO
IdentifierType	string	example.: IMEI / SNR / MAC / etc	Y	1
IdentifierValue	string		Y	2

Messagetype: BaseToken

Field	Type	Description	M	SO
TokenType	string	MFA, GST, EPAN, UID …	Y	1
TokenValue	string		Y	2

Propertybag	Propertybag		N	-

Messagetype: Propertybag, collection of keyvaluepairs.

Field	Type	Description	M	SO
Key	string		Y	-
Value	Object	Basic types are supported like, bytes, integers, string etc.	Y	-

Messagetype: BaseService

Field	Type	Description	M	SO
ServiceId	long		Y	1

Messagetype: ServiceRequestData

Field	Type	Description	M	SO
RequestExternalIpAddress	string	internal IP-Address of sensing device (format: 2.2.2.2)	N	1
RequestInternalIpAddress	string	internal IP-Address of sensing device (format: 2.2.2.2)	N	2
RequestSensorLocalTimestamp	DateTime	Local timestamp	Y	3
Amount	Integer		Y	4
CurrencyCode	String		Y	5
RequestMode	integer	1 = Online 2 = StoreAndForward 4 = Offline	Y	6
PropertyBag	PropertyBag		N	-

Return messagetype: ResponseMessage

Field	Type	Description
Transaction	Transaction	Identical structure and values to the request.
Message	String	Optional message
ResponseValue	Int	
Propertybag	Propertybag	

Signing a message

We sign the messages with the following procedure.
Used encoding: "ISO-8859-1"
DateTime values are presented via the UNIX timestamp.
Guids are presented in the following format: '00000000-0000-0000-0000-000000000000'

We concatenate per field the fieldlength (max value 999999) + the fieldvalue
We concatenate all fields based on the order mentioned in the models above.
Signature is created based the binary representation of the field concatenation mentioned above with a SHA256 hash using the private key of the used certificate.

Error codes

If a sensor api call fails, the api returns an errorcode based on the standardized HTTP error codes
See https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Used error codes:

Error code	Description / Example
400	Bad request – Invalid model (data send to the API does not meet minimal requirements)
400	Bad request - serviceId needs to be a positive number.
400	Bad request – invalid sensor (sensorid is unknown)
400	Bad request – no certificate available (signature is set, but certificate is not known for this sensorid)
400	Bad request – no signature available (no signature is set, but is required by sensorconfiguration)
400	Bad request – invalid signature (signature does not match the data provided)
500	Internal server error – general error.

5.1. Mobile API

The Mobile API is used to connect the EcoSpace Hub to thirdparty mobile app-builders. The mobile API supports multiple calls with a user centric approach.

Account

Account centric calls

Create Account

Request messagetype: CreateUserModel

Field	Type	Description
Email	String	
Password	string	
ConfirmPassword	string	

Return messagetype: CreateUserResponseModel

Field	Type	Description
Success	Bool	
Response	Object	Complex type with extra information if user account creation failed.

Login

Login is based on OAuth2 specs.
Following parameters are required

Field	Type	Description
Username	String	
Password	String	
Grand_type	String	Fixed value = password
Client_id	String	
Device_Id	String	Optional, identify device when logged in on multiple devices
Client_Secret	String	

Register device for push notifications

Request messagetype: RegisterDeviceModel

Field	Type	Description
RegistrationId	String	
PlatformType	int	Android =0, WindowsPhone =1 , IOS =2

Response: HttpCode: 200 or 500

UnRegister device for push notifications

Request messagetype: UnRegisterDeviceModel

Field	Type	Description
RegistrationId	String	

Response: HttpCode: 200 or 500

Load Profile

Response message: UserProfileResult

Field	Type	Description
Success	Bool	
Remark	String	
Result	UserProfile	

UserProfile

Field	Type	Description
FirstName	String	
MiddelName	String	
LastName	String	
DateOfBirth	DateTime	
MobilePhoneNumber	String	
Address	String	
ZipCode	String	
BankAccount	String	
SWITFCode	String	

City	String	
Country	String	

Save Profile

Request message: UserProfile

Field	Type	Description
FirstName	String	
MiddelName	String	
LastName	String	
DateOfBirth	DateTime	
MobilePhoneNumber	String	
Address	String	
ZipCode	String	
BankAccount	String	
SWITFCode	String	
City	String	
Country	String	

Paymentservices

Get available paymentservices

Response collection of paymentmodels
Model: paymentmodels

Field	Type	Description
Id	long	
Name	String	
Displayname	String	
Content	Dictionary with icon urls	
RequiredUserData	Collection of strings Collection of profile categories	Optional, contains the required userprofile information that is shared with the payment method when the paymentmethod is enabled for the user.
Enabled	Bool	
TokenValue	String	Optional
TokenTypeId	Int	Optional – default = 0
OrderId	Int	Optional – default = 0

Get enabled paymentservices for token

Request: tokenvalue as string

Response: collection of PaymentModel (see get available paymentservices)

Get specific paymentmethod for token

Request: tokenvalue (string), paymentmethod (string), currencycode (string)

Response: PaymentModelDetails

Field	Type	Description
Id	long	
Name	String	
Displayname	String	
Content	Dictionary with icon urls	
RequiredUserData	Collection of strings Collection of profile categories	Optional, contains the required userprofile information that is shared with the payment method when the paymentmethod is enabled for the user.
Enabled	Bool	
TokenValue	String	Optional
TokenTypeId	Int	Optional – default = 0
OrderId	Int	Optional – default = 0
CurrencyCode	String	
Value	Long	Amount (in case of prepaid paymentmethod).
Valid	Bool	Value is valid, in case of postpaid valid is false.

Enable payment method for token

Request: tokenvalue (string), paymentmethod (string), currencycode (string)
Request: tokenvalue (string), paymentmethod (string), currencycode (string)

Prioritize payment methods for tokens

Request: tokenvalue as string & array of paymentserviceIds.
Remark: order of the array contains order of the paymentmethods Response: httpcode 202 (Accepted) or 409 (Conflict)

Pouch

Get all tokens in Pouch

Request: none
Response model: Collection of PouchItem

Field	Type	Description
RegisteredTo	String	(a.k.a. email)
EngravedId	String	
TokenTypeId	Int	
TokenValue	String	
Description	String	Optional user description of the

		token)
FormFactor	String	
TravelSchemeName	String	
ActivationCode	String	

Get single token in Pouch

Request: tokenvalue (string)
Response model: PouchItem

Add Token to Pouch

Request: PouchItem
Response: httpcode 201 (Created) or 400 (Badrequest) with extra error information.

Services

Get all available Services

Request: none
Response Collection of ServiceModel
Model: ServiceModel

Field	Type	Description
ServiceId	long	
ServiceName	string	
ParentHubName	String	Service can be related with a foreignhub, if so the foreignhub is mentioned
TokenValue	String	Optional
TokenTypeId	Int	Optional, defaultvalue = 0

Get all enabled services for token

Request: tokenvalue
Response Collection of ServiceModel

Subscribe to service for token

Request UpdateSubscriptionModel
Model UpdateSubscriptionModel

Field	Type	Description
ServiceId	long	
ServiceName	string	
ParentHubName	String	Service can be related with a foreignhub, if so the foreignhub is mentioned
TokenValue	String	
TokenTypeId	Int	

Response: httpcode 201 (Created) or 400 (Badrequest) with extra error information.

Unsubscribe to service for token

Request UpdateSubscriptionModel
Model UpdateSubscriptionModel

Field	Type	Description
ServiceId	long	
ServiceName	string	
ParentHubName	String	Service can be related with a foreignhub, if so the foreignhub is mentioned
TokenValue	String	
TokenTypeId	Int	

Response: httpcode 201 (Created) or 400 (Badrequest) with extra error information.

Vouchers

Get all available vouchers

Request: none
Response: collection of TicketVoucher
Model TicketVoucher

Field	Type	Description
ExternalId	Guid	
Description	string	
Enabled	Bool	Is the voucher already redeembed
ValidFrom	DateTime	Voucher can be redeemed from
validUntil	DateTime	Voucher can be redeemed until
VoucherType	String	
RedeemedDateTime		When was the voucher redeemed
BoughtWithTokenValue	String	

Get voucher detail

Request: id (guid)
Response TicketVoucher

Redeem voucher

Request: id (guid)
Response: httpcode 202 (Accepted) of 406 (Not acceptable)

VReceipt

Get all Vreceipts

Request datetime (format yyyyMMddhhmmss) (optional)
Response VirtualReceiptItemContainer

Model: VirtualReceiptItemContainer

Field	**Type**	**Description**
PaymentMethods	Dictionary	Key = Id, value = displayname
Statuses	Dictionary	Key = Id, value = displayname
Types	Dictionary	Key = Id, value = displayname
CurrencyCodes	Dictionary	Key = Id, value = displayname
ServiceTokens	Dictionary	Key = Id, value = tokenValue
Items	Collection of VirtualReceiptItems	

Model: VirtualReceiptItem

Field	**Type**	**Description**
Id	Guid	
e	string	ExternalTransactionId
a	Long	Amount
c	Int	CurrencyCode (look up key for dictionary)
p	int	PaymentMethod (look up key for dictionary)
r	DateTime	ReceiptDate
s	Int	ServiceToken (look up key for dictionary)
u	Int	Status (look up key for dictionary)
t	Int	Type (look up key for dictionary)
Items	Collection of VirtualReceiptLineItem	Optional

Model: VirtualReceiptLineItem

Field	**Type**	**Description**
Id	Int	
VirtualReceiptKey	String	
VATPercentage	Decimal	Optional
VAT	Int	Optional
Total	Int	Optional
Price	Int	Optional

NumberOfItems	Int	Optional
ExtraInformation	String	
Description	String	
DateTime	String	

Get VReceipt Details

Request: transactionid (Guid)
Response VirtualReceiptItemContainer

Errorcodes

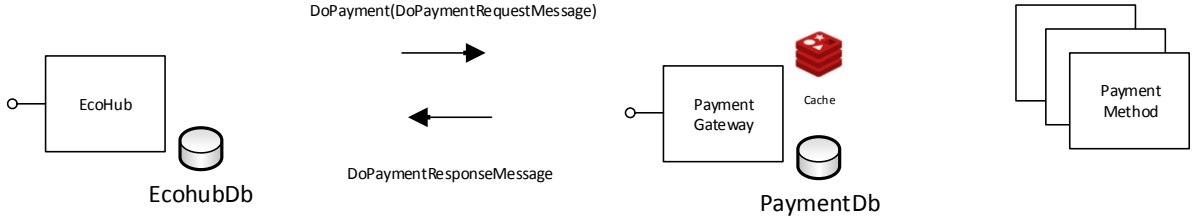
Error code	**Description / Example**
400	Bad request – Invalid model (data send to the API does not meet minimal requirements)
400	Bad request - serviceId needs to be a positive number.
400	Bad request – invalid sensor (sensorid is unknown)
400	Bad request – no certificate available (signature is set, but certificate is not known for this sensorid)
400	Bad request – no signature available (no signature is set, but is required by sensorconfiguration)
400	Bad request – invalid signature (signature does not match the data provided)
500	Internal server error – general error.
409	Conflict – Object we try to create already exists. (example: en account with known emailaddress)

5.1. Payment Gateway

The PaymentGateway accepts the following commands which should be implemented by connected paymentmethods:

Method	**M**
DoPayment	Y
ReserveCredit	Y
CancelPayment	Y
GetBalance	N
TopUp	N
SpendCeiling	N
Refund	N

Connected payment methods have to implement the mandatory methods, and can implement the optional methods.



DoPayment

DoPaymentRequestMessage

Field	Type	Description	M	SO
ServiceToken	ServiceToken		Y	-
SensorId	Guid	Identifier of the Sensor	N	-
Amount	Integer	Amount in cents, could be 0	Y	-
TapTimestamp	DateTime		Y	-
ExternalTransactionId	String		Y	-
CurrencyCode	String	International currency code	Y	-
AcceptedPaymentMethods	List (of: String)	If this list is empty, any paymentmethod is accepted. If not, only paymentmethods in the list can be tried.	N	-

DoPaymentResponseMessage (Inherits ResponseMessage)

Field	Type	Description	M	SO
TransactionId	String	TransactionId of PaymentMethodGateway	Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
TransactionStatus	Enum	1=SUCCESS, 2=UNKNOWN_CLIENTID, 3=UNKNOWN_READERID, 4=INSUFFICIENT_BALANCE, 5=TIMEOUT, 6=GENERAL_ERROR	Y	-

ResponseMessages:

Message	ResponseValue	Description
SUCCESS	>= 0	DoPaymentRequest was successfully handled by PaymentGateway.
NO PAYMENTSUBSCRIPTION FOUND FOR SERVICETOKEN	-4	No matching available paymentmethod was found. (Match was tried for available paymentmethods of token-owner and

		supported paymentmethods of merchant)
NO ACTIVE PAYMENTSUBSCRIPTIONS FOUND FOR SERVICETOKEN	-3	Registered token has no active paymentsubscriptions (= active paymentmethods) available in paymentgateway.
<General Error>	-1	Message contains the error.

ReserveCredit

ReserveCreditRequestMessage

Field	Type	Description	M	SO
ServiceToken	ServiceToken		Y	-
SensorId	Guid	Identifier of the Sensor	N	-
Amount	Integer	Amount in cents, could be 0	Y	-
TapTimestamp	DateTime		Y	-
ExternalTransactionId	String		Y	-
CurrencyCode	String	International currency code	Y	-
AcceptedPaymentMethods	List (of: String)	If this list is empty, any paymentmethod is accepted. If not, only paymentmethods in the list can be tried.	N	-

ReserveCreditResponseMessage

Field	Type	Description	M	SO
TransactionId	String	TransactionId of PaymentMethodGateway	Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
TransactionStatus	Enum	1=SUCCESS, 2=UNKNOWN_CLIENTID, 3=UNKNOWN_READERID, 4=INSUFFICIENT_BALANCE, 5=TIMEOUT, 6=GENERAL_ERROR	Y	-

CancelPayment

Cancels payment reservation

CancelPaymentRequestMessage

Field	Type	Description	M	SO
TransactionId	String		Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-

CancelPaymentResponseMessage

Field	Type	Description	M	SO
TransactionId	String	TransactionId of PaymentMethodGateway	Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
TransactionStatus	Enum	1=SUCCESS, 2=UNKNOWN_CLIENTID, 3=UNKNOWN_READERID, 4=INSUFFICIENT_BALANCE, 5=TIMEOUT, 6=GENERAL_ERROR	Y	-

GetBalance

GetBalanceRequestMessage

Field	Type	Description	M	SO
ServiceToken	ServiceToken		Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-

GetBalanceResponseMessage

Field	Type	Description	M	SO
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
WalletId	String	Identification of wallet in PaymentMethod	Y	-
Balance	Integer	1=SUCCESS, 2=UNKNOWN_CLIENTID, 3=UNKNOWN_READERID, 4=INSUFFICIENT_BALANCE, 5=TIMEOUT, 6=GENERAL_ERROR	Y	-
CurrencyCode	String	International currency code		

TopUp

TopUpRequestMessage

Field	Type	Description	M	SO
ServiceToken	ServiceToken		Y	-
SensorId	Guid	Identifier of the Sensor	N	-

Amount	Integer	Amount in cents, could be 0	Y	-
TapTimestamp	DateTime		Y	-
ExternalTransactionId	String		Y	-
CurrencyCode	String	International currency code	Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-

TopUpResponseMessage (Inherits ResponseMessage)

Field	Type	Description	M	SO
TransactionId	String	TransactionId of PaymentMethodGateway	Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
TransactionStatus	Enum	1=SUCCESS, 2=UNKNOWN_CLIENTID, 3=UNKNOWN_READERID, 5=TIMEOUT, 6=GENERAL_ERROR	Y	-

ResponseMessages:

Message	ResponseValue	Description
SUCCESS	>= 0	DoPaymentRequest was successfully handled by PaymentGateway.
NO PAYMENTSUBSCRIPTION FOUND FOR SERVICETOKEN	-4	No matching available paymentmethod was found. (Match was tried for available paymentmethods of token-owner and supported paymentmethods of merchant)
NO ACTIVE PAYMENTSUBSCRIPTIONS FOUND FOR SERVICETOKEN	-3	Registered token has no active paymentsubscriptions (= active paymentmethods) available in paymentgateway.
<General Error>	-1	Message contains the error.

SpendCeiling

SpendCeilingRequestMessage

Field	Type	Description	M	SO
ServiceToken	ServiceToken		Y	-

PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
CeilingType	Integer	Day, Week, Month, Year or Period Ceilings	N	-

SpendCeilingResponseMessage

Field	Type	Description	M	SO
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
Balance	Integer	1=SUCCESS, 2=UNKNOWN_CLIENTID, 3=UNKNOWN_READERID, 4=INSUFFICIENT_BALANCE, 5=TIMEOUT, 6=GENERAL_ERROR	Y	-
CurrencyCode	String	International currency code		

Refund

RefundRequestMessage

Field	Type	Description	M	SO
ServiceToken	ServiceToken		Y	-
SensorId	Guid	Identifier of the Sensor	N	-
Amount	Integer	Amount in cents, could be 0	Y	-
TapTimestamp	DateTime		Y	-
ExternalTransactionId	String		Y	-
CurrencyCode	String	International currency code	Y	-
RefundTransactionReference	String	Reference to PaymentTransaction for which a refund is applied	Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-

RefundResponseMessage (Inherits ResponseMessage)

Field	Type	Description	M	SO
TransactionId	String	TransactionId of PaymentMethodGateway	Y	-
PaymentMethod	PaymentMethod	See **Fout! Verwijzingsbron niet gevonden.**	Y	-
TransactionStatus	Enum	1=SUCCESS, 2=UNKNOWN_CLIENTID, 3=UNKNOWN_READERID, 5=TIMEOUT, 6=GENERAL_ERROR	Y	-

ResponseMessages:

Message	ResponseValue	Description
SUCCESS	>= 0	DoPaymentRequest was successfully handled by PaymentGateway.
NO PAYMENTSUBSCRIPTION FOUND FOR SERVICETOKEN	-4	No matching available paymentmethod was found. (Match was tried for available paymentmethods of token-owner and supported paymentmethods of merchant)
NO ACTIVE PAYMENTSUBSCRIPTIONS FOUND FOR SERVICETOKEN	-3	Registered token has no active paymentsubscriptions (= active paymentmethods) available in paymentgateway.
<General Error>	-1	Message contains the error.

Payment Method Stub Interface

A paymentmethod-stub is a component that links an exisiting external paymentmethod with the payment gateway. A paymentmethod-stub should implement all mandatory methods mentioned in 5.1
Optional methods can be implemented, each connected payment-method should register at the paymentgateway with the correct paymentmethod-settings.

6. List of Deliverables – Grant Agreement

1.3.2. WT2 list of deliverables

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D1.1	Inception Report	WP1	1 - Stichting Open Ticketing	Report	Public	2
D1.2	Program Plan	WP1	1 - Stichting Open Ticketing	Report	Public	3
D1.3	Final Report	WP1	1 - Stichting Open Ticketing	Report	Public	21
D1.4	Close-out Report	WP1	1 - Stichting Open Ticketing	Report	Public	24
D1.5	Online Portal	WP1	1 - Stichting Open Ticketing	Demonstrator	Public	6
D2.1	Master Test Plan	WP2	5 - UL TS BV	Report	Public	9
D2.2	Test Case Specifications	WP2	5 - UL TS BV	Report	Public	9
D2.3	Test Report	WP2	5 - UL TS BV	Report	Public	13
D3.1	ETC statutes	WP3	1 - Stichting Open Ticketing	Report	Public	3
D3.2	Ecosystem ABT	WP3	1 - Stichting Open Ticketing	Report	Public	5
D3.3	Franchise Framework	WP3	1 - Stichting Open Ticketing	Report	Public	12
D3.4	ETC Business Plan	WP3	1 - Stichting Open Ticketing	Report	Public	13
D3.5	Business Case ABT	WP3	1 - Stichting Open Ticketing	Report	Public	14
D3.6	Franchise Contracts	WP3	1 - Stichting Open Ticketing	Report	Public	23
D4.1	Privacy Reference Design	WP4	1 - Stichting Open Ticketing	Other	Public	6
D4.2	Smart Privacy Requirements	WP4	1 - Stichting Open Ticketing	Report	Public	9
D4.3	Audit Procedures	WP4	1 - Stichting Open Ticketing	Other	Public	12
D5.1	Lab Expansion Plan	WP5	1 - Stichting Open Ticketing	Report	Public	2
D5.2	Staging Server	WP5	1 - Stichting Open Ticketing	Demonstrator	Public	3
D5.3	Showcase Lab	WP5	1 - Stichting Open Ticketing	Demonstrator	Public	9
D6.1	Hub Design	WP6	1 - Stichting Open Ticketing	Report	Public	3

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D6.2	Travel Scheme Hub	WP6	1 - Stichting Open Ticketing	Other	Confidential, only for members of the consortium (including the Commission Services)	6
D6.3	Interoperability Hub	WP6	1 - Stichting Open Ticketing	Other	Confidential, only for members of the consortium (including the Commission Services)	10
D6.4	License Agreement	WP6	1 - Stichting Open Ticketing	Report	Public	12
D6.5	Open Standards & Requirements	WP6	1 - Stichting Open Ticketing	Other	Public	12
D7.1	Interface Specification Document	WP7	5 - UL TS BV	Other	Public	7
D7.2	Test Suite	WP7	5 - UL TS BV	Demonstrator	Public	9
D7.3	GST implementation on NXP platform	WP7	3 - NXP Semiconductors	Demonstrator	Confidential, only for members of the consortium (including the Commission Services)	10
D7.4	GST implementation on Java Card Platform	WP7	5 - UL TS BV	Demonstrator	Public	10
D7.5	Test report GST	WP7	5 - UL TS BV	Report	Public	13
D7.6	Low-cost Implementation Strategy	WP7	1 - Stichting Open Ticketing	Report	Public	13
D8.1	Functional Specifications Interoperable Account System	WP8	1 - Stichting Open Ticketing	Report	Public	6
D8.2	Demo-System Interoperable Account System	WP8	1 - Stichting Open Ticketing	Demonstrator	Public	9
D8.3	Open Standards & Requirements for IAS	WP8	1 - Stichting Open Ticketing	Other	Public	12
D9.1	Functional Specifications Interoperable Travellers Interface (ITI)	WP9	1 - Stichting Open Ticketing	Report	Public	6

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D9.2	Demo System ITI	WP9	1 - Stichting Open Ticketing	Demonstrator	Public	9
D9.3	Open Standards & Requirements ITI	WP9	1 - Stichting Open Ticketing	Other	Public	12
D10.1	Pilot Design Document	WP10	1 - Stichting Open Ticketing	Report	Public	4
D10.2	Project Plan Pilots	WP10	1 - Stichting Open Ticketing	Report	Public	9
D11.1	ABT systems & equipment	WP11	6 - Trans Link Systems	Demonstrator	Public	9
D11.2	Mobile Travel App	WP11	6 - Trans Link Systems	Demonstrator	Public	11
D11.3	Acceptance of the Dutch Account in Germany	WP11	6 - Trans Link Systems	Demonstrator	Public	13
D11.4	Acceptance of the German Account in the Netherlands	WP11	6 - Trans Link Systems	Demonstrator	Public	15
D11.5	Introduction of GST	WP11	6 - Trans Link Systems	Demonstrator	Public	17
D12.1	Mobility Info	WP12	2 - ETS	Report	Public	3
D12.2	Reservation & Ticketing Platform	WP12	2 - ETS	Demonstrator	Public	13
D12.3	Fare Medium	WP12	2 - ETS	Demonstrator	Public	12
D12.4	ABT equipment adaptation	WP12	2 - ETS	Demonstrator	Public	12
D12.5	Pilot	WP12	2 - ETS	Demonstrator	Public	14
D12.6	Standards & Requirements for Germany	WP12	2 - ETS	Other	Public	11
D13.1	Pilot Luxembourg	WP13	4 - COMMUNAUTE DES TRANSPORTS	Demonstrator	Public	14
D13.2	Pilot Report Luxembourg	WP13	4 - COMMUNAUTE DES TRANSPORTS	Report	Public	20
D14.1	Test & Evaluation Plan	WP14	5 - UL TS BV	Report	Public	9
D14.2	End-to-End Test Report	WP14	5 - UL TS BV	Report	Public	21

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D14.3	Pilot Report	WP14	1 - Stichting Open Ticketing	Report	Public	21