ACCEPT Institute

Horizon 2020 ETC 636126

Open Standards & Requirements

–

Deliverable 6.5

20 April 2018

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 636126.

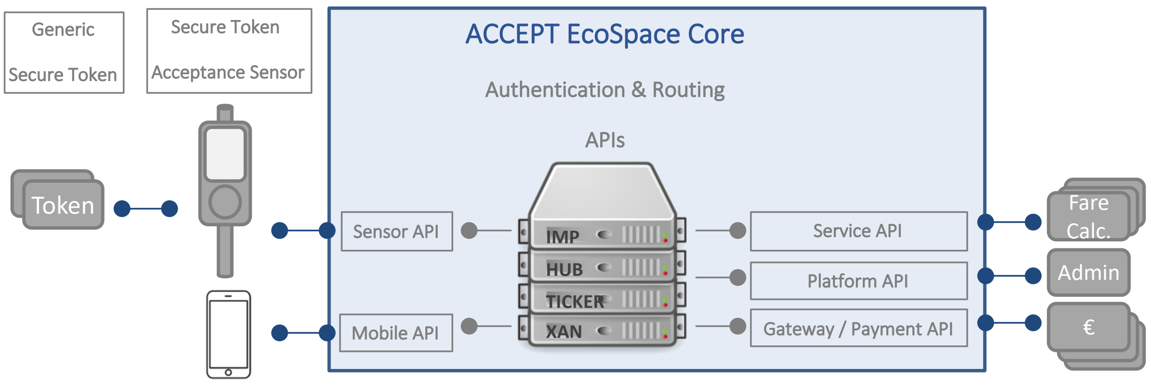
1 Introduction

1.1 Introduction & Summary

This document provides an overview of the Open Standards & Requirements for the ACCEPT Eco Space Core. The standards and requirements for the ETC Hubs (ACCEPT EcoSpace Core Software) will be published in order to allow ETC Members to develop their own local or regional hubs. Part of this deliverable is to provide the complete overview and to draft the Sensor API.

In chapter 4 of this document an overview is provided of all open standards. We have indicated in which deliverable of the ETC project these standards and requirements can be found, as they are not included in a single deliverable. Part of this deliverable is the so-called Sensor API, an API that is to be used in order to connect a Secure Token Acceptance Sensor (eg. validator, gate) to the ACCEPT EcoSpace Core (see figure below).

The different standards & requirements are:



-	ACCEPT EcoSpace Core	See deliverable 6.1 and deliverable 6.4
-	Sensor API	See appendix A to this document
-	Mobile API	See deliverable 9.3
-	Service API	See deliverable 8.3
-	Platform API	See deliverable 8.3
-	Gateway / Payment API	See deliverable 8.3
-	Secure Token Acceptance Sensor	See deliverable 7.1

-	Generic Secure Token	See deliverable 7.4

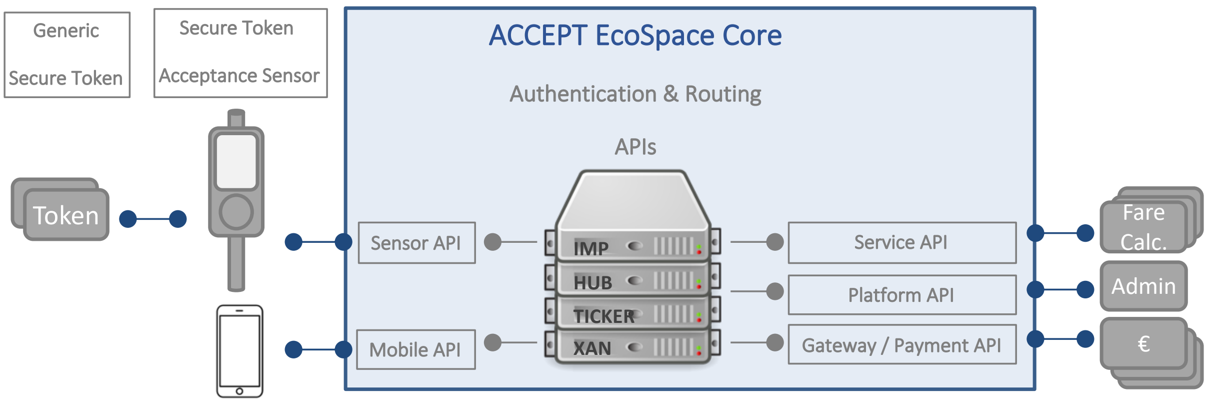
Deliverable 6.5 is part of work package 6 ‘*Define & Develop Authentication & Routing Hub & Token ID management*’.

2 Content

3 ACCEPT EcoSpace Core Software

3.1 ECO Space Core Software

Below a high-level description of the ACCEPT EcoSpace Core Software is presented. It starts with an overview of the logical components, followed by the overview of the APIs and the micro-services.



3.2 Logical Components

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.

- 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.

3.3 APIs

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 (*secure token acceptance sensor*[1]) 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.

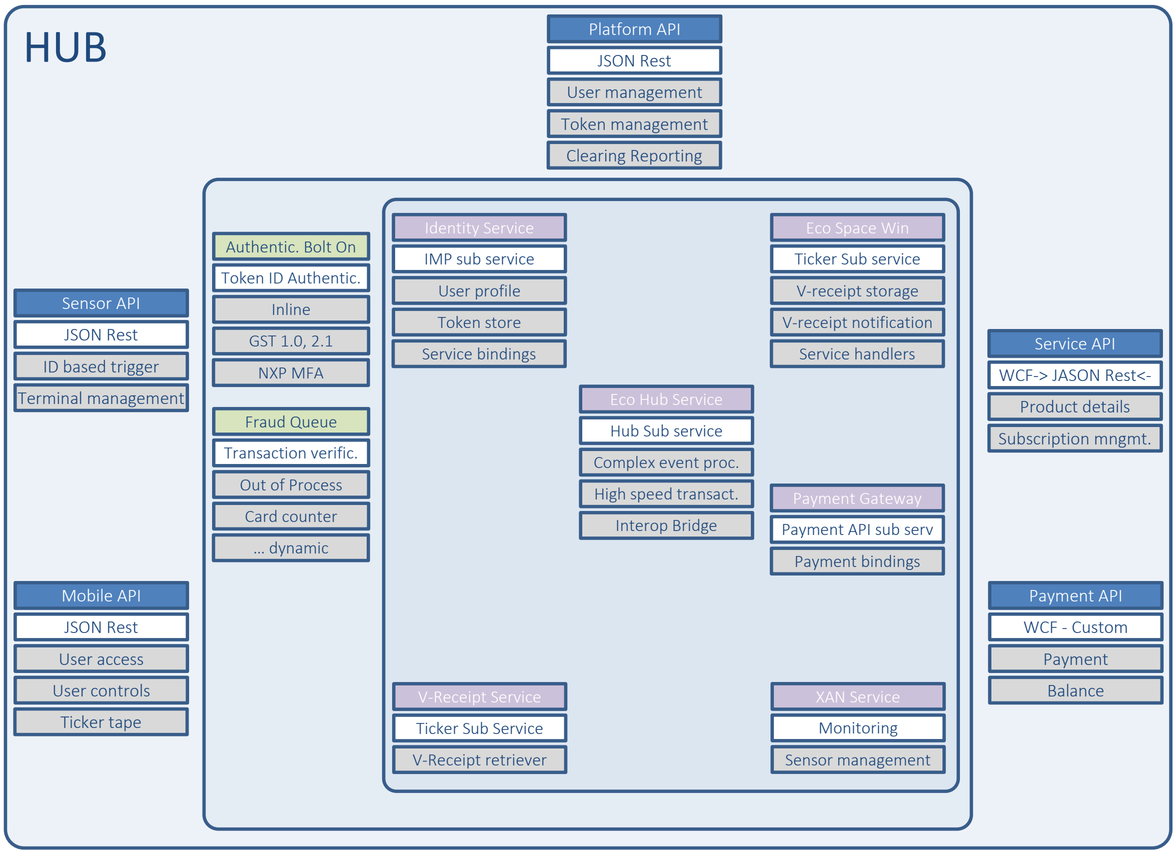
- GATEWAY / PAYMENT API:
  High speed secure router for normalizing custom developed bank connections to a universally consumable payment method.

- PLATFORM API:
  To afford users of the Platform API (owners of a particular Hub) access to administrative functions related to Account Creation, Token Registration, and Transaction Reporting.

3.4 Micro Services

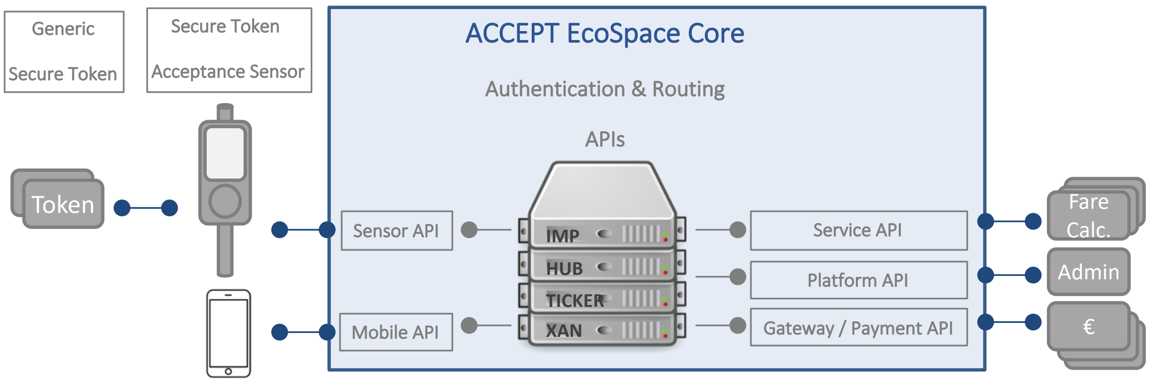
Logical components such as IMP, HUB, Ticker and XAN, are comprised of underlying **micro services**.

[1] The specification of the necessary software in the Secure Token Acceptance Sensor (STAS) is described in the document: "Secure Token Acceptance Sensor *Behavior and Interface Specification*). This document provides the interface specification between the STAS and the ACCEPT EcoSpace Core and the interface between the STAS and the Generic Secure Token (GST) ID.


HUB
Platform API
JSON Rest
User management
Token management
Clearing Reporting
Sensor API
JSON Rest
ID based trigger
Terminal management
Mobile API
JSON Rest
User access
User controls
Ticker tape
Authentic. Bolt On
Token ID Authentic.
Inline
GST 1.0, 2.1
NXP MFA
Fraud Queue
Transaction verific.
Out of Process
Card counter
... dynamic
Identity Service
IMP sub service
User profile
Token store
Service bindings
Eco Space Win
Ticker Sub service
V-receipt storage
V-receipt notification
Service handlers
Eco Hub Service
Hub Sub service
Complex event proc.
High speed transact.
Interop Bridge
Payment Gateway
Payment API sub serv
Payment bindings
V-Receipt Service
Ticker Sub Service
V-Receipt retriever
XAN Service
Monitoring
Sensor management
Service API
WCF-> JASON Rest<-
Product details
Subscription mngmt.
Payment API
WCF - Custom
Payment
Balance

4 Open Standards & Requirements

The different standards & requirements are:



- ACCEPT EcoSpace Core — See deliverable 6.1 and deliverable 6.4
  The ACCEPT EcoSpace Core is the central component, a so-called Authentication & Routing Hub. The Hub authenticates and routes transactions and consists of the following components:
  - IMP: Identity Management Platform for Private, flexible and intuitive Account Creation.
  - 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.

- Sensor API — See appendix A to this document
  The Sensor API is used to connect front end devices to the ACCEPT EcoSpace Core. Examples of front end devices are gates (on stations), or validators (in busses). In technical terms these devices are called a Secure Token Acceptance Sensor. The software necessary for the devices to accept the Generic Secure Token (which is used in the ETC pilots as the identifier) is described in deliverable 7.1.

- Mobile API — See deliverable 9.3
  The Mobile API is used to connect smartphone apps (or traveller interfaces) to the ACCEPT EcoSpace Core. Through this API is it possible to transfer transactions from the ACCEPT EcoSpace Core to the smartphone app.

- Service API — See deliverable 8.3
  The Service API is part of the API's used by the different account systems to connect to the ACCEPT EcoSpace Core. Examples of services are: online ticket stock, where public transport tickets are stored, used for travel.

- Platform API — See deliverable 8.3

The Platform API is part of the API’s used by different account systems to connect to the ACCEPT EcoSpace Core. It can be used to access to administrative functions related to Account Creation, Token Registration, and Transaction Reporting.

- Gateway / Payment API	See deliverable 8.3
  The Gateway or Payment API is part of the API’s used by different account systems to connect to the ACCEPT EcoSpace Core. The API can be used to connect to a universally consumable payment method.

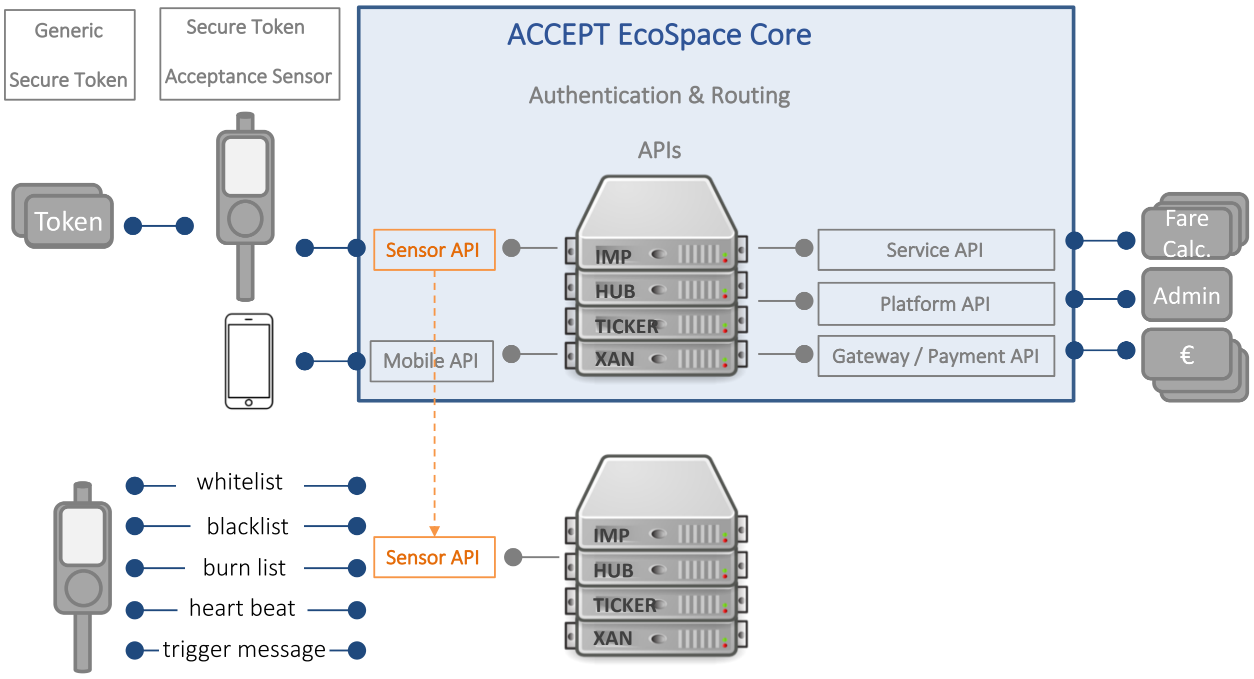
- Secure Token Acceptance Sensor	See deliverable 7.1
  The Secure Token Acceptance Sensor (STAS) is the front-end device used in the ETC project. Examples are validators in busses, or gates at parking locations. In order to be able to communicate with the Generic Secure Token, the STAS needs software that is described in deliverable 7.1.

- Generic Secure Token	See deliverable 7.4
  The Generic Secure Token (GST) is a side token that is used in the ETC pilots and resides on a smartcard. The GST is used as the identifier for a traveller. Specifications of the GST are described in deliverable 7.4 and can be used by card manufacturers to implement the GST.

Appendix A: Sensor API

The Sensor API is used to connect sensors, e.g. contactless readers (or NFC readers) to the 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.



Message Overview

There are 5 post messages that a sensor can submit to the EcoSpace Core via the Sensor API:

chIMP Messages

1. Whitelist
2. Blacklist
3. Burnlist

XAN Messages

4. Heartbeat

Hub Messages

5. Trigger Message

chIMP Messages

For EcoSpaces that need to support off-line or non-real-time use cases, the Sensors can store relevant portions of the IMP registry locally (in the chIMP). The Token IDs in a chIMP message are never transferred in the clear. They are salted and hashed and the local NFC Reader must run the Token ID it sees through the same process to match it to a card to in the locally stored a hashed list. These messages are:

- White List
- Black List
- Burn List

XAN Messages

Sensors are asked to send heartbeats to the XAN to ensure that quality of service for operations in the field can be monitored and displayed in the Sysops Dashboard feeds. The Heartbeat response is also used to update Terminals.

- Heartbeat

Hub Message

The main job of the sensor is to send Trigger messages when tapped.

- Trigger Message

Sensor API Web Page

The Sensor API for the ACCEPT Test Environment also has a public API Web Page, to help developers with reference code examples and templates and web forms for submission of code samples to allow developers to test drive of their own code. While the page is public there are certain certificates and permissions required before developers can set up and test their own code.

Sensor API Web Page for Test Environment

https://oti-test-nlhub.westeurope.cloudapp.azure.com/Sensor/swagger/ui/index

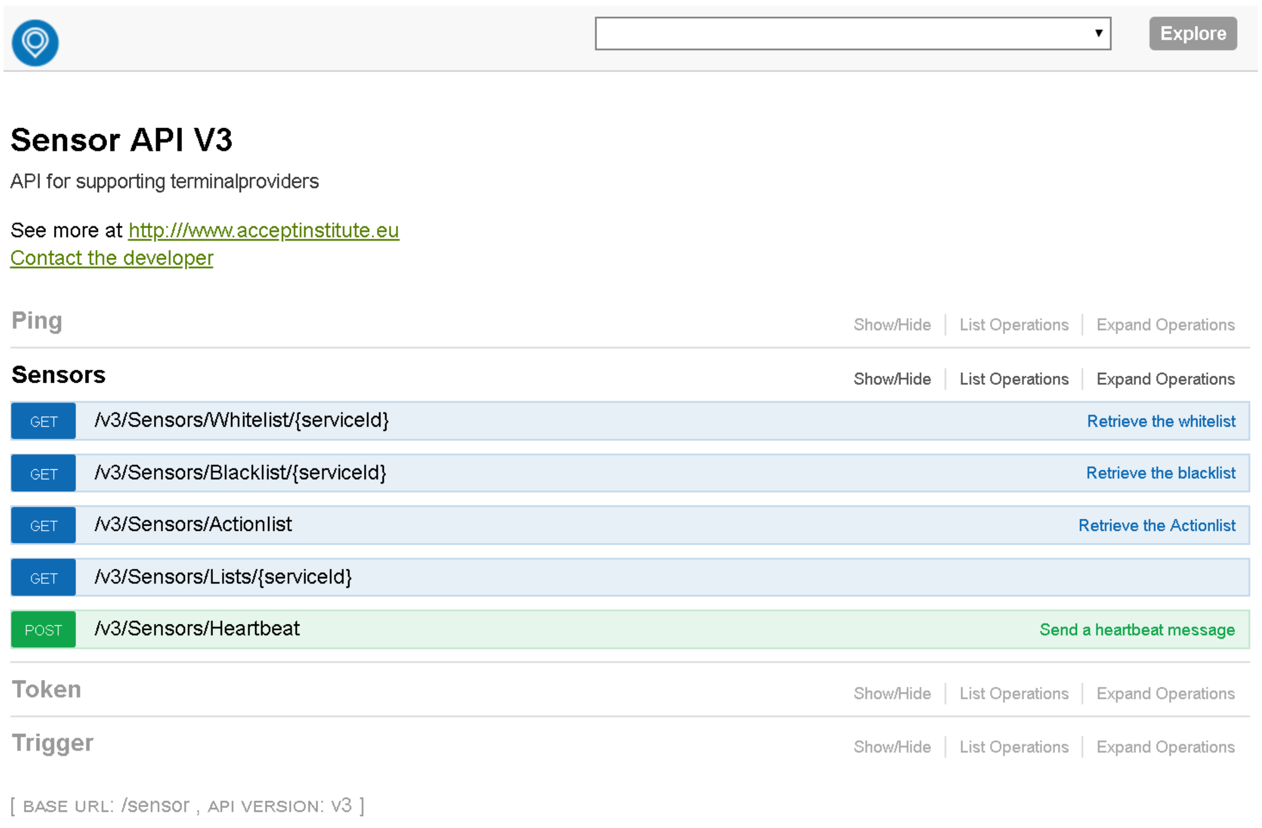
Message Formats

XML or JSON

Message Transport

HTTPS REST

Note: Right now, HTTPS certs are self-generated by the receiving web server.



Sensor API V3

API for supporting terminalproviders

See more at http://www.acceptinstitute.eu
Contact the developer

Ping — Show/Hide | List Operations | Expand Operations

Sensors — Show/Hide | List Operations | Expand Operations

GET	/v3/Sensors/Whitelist/{serviceId}	Retrieve the whitelist
GET	/v3/Sensors/Blacklist/{serviceId}	Retrieve the blacklist
GET	/v3/Sensors/Actionlist	Retrieve the Actionlist
GET	/v3/Sensors/Lists/{serviceId}	
POST	/v3/Sensors/Heartbeat	Send a heartbeat message

Token — Show/Hide | List Operations | Expand Operations

Trigger — Show/Hide | List Operations | Expand Operations

[BASE URL: /sensor , API VERSION: v3]

Sensor Onboarding & Configuration

When a Sensor is introduced to an EcoSpace, multiple steps take place to attach it to the EcoSpace. Typical installation includes:

Pre-Registration

Prior to installation the Sensor should be pre-registered for the EcoSpace Production or Test Environment it is intended to operate in.

- Sensor PKI Certificate is shared with the EcoSpace.
- An ACCEPT PKI Certificate is shared for integration in the Sensor for Off-line use cases.

- Service Hosting Selection – Refers to which service(s) it will host; directly related to its ability to create the appropriate Trigger Message for that particular Service.
- Payment Method Acceptance Capabilities – Which Payment Method.
- Sensor Token Capabilities - Such as Which tokens it supports.
- Sensor Operation Modes - if it will operate online, offline or partial.
- Location Information – GPS, CellID.
- Upon preregistration a Sensor ID is issued.

Note: Pre-registration is a manual process, please contact ACCEPT for pre-registration of test and pilot sensors (terminals).

Activation

Once a Sensor is ready for testing or placement it can be activated by attaching it to the EcoSpace Environment.

Upon Placement, the Sensor, which has been pre-configured to send heartbeats to the EcoSpace XAN, will receive a challenge consisting of a short code for multifactor activation of the Sensor on the EcoSpace network. This requires that the installation technician or the merchant in question uses the Sensor Installation Mobile App.

Documents (pdf.-files) Table of Content:

1. Aggregate List V3

Aggregate List V3

Retrieve Aggregate List

Retrieves a list of ETC Token IDs that are white or black listed listed for a given service hosted by the STAS, or are on the ActionList

Use API Get: /V3/Sensors/Lists/{serviceId}

where serviceId is the serviceId (type Long) for a whitelist to retrieve

With no message body

Expected response

Field	Type	Description	M
Data	SensorActionListResponse	object	Y
Success	Bool	True = Success	Y
Message	String	Optional extra details regarding results	N

SensorActionListResponse

Field	Type	Description / Value
List	Object	List of ListItem objects, described in 5.3.2.1

The list is binary ordered on the TokenHash

Data Object ListItem

Field	Type	Description / Value
TokenHash	String	Base64 encoded hash of TokenID, see section 5.2
TokenType	String	Token Type and version. “GST21” for GST 2.x (GSTVersion 0x02 00 – 0x02 FF)
ListType	String	W for whitelist, B for Blacklist, empty for no value.
ActionList	Array	List of ActionListItems. See 5.3.2.2

ActionlistItem for GST

Field	Type	Description / Value
ActionType	String	Type of action. “APDU” for GST
APDUValue	String	Hexadecimal representation of binary APDU

2. Message Signing Sensor API V3

Message Signing Sensor API V3

V3 Signature Creation and validation for incoming Api-requests and outgoing Api-responses

For all V3-incoming-api calls signature creation and -validation is done using the following procedure:

The following http-headers must be added to the message:

- SensorID - contains the hub-assigned sensorid (format is GUID)

if a sensor is configured to use a certificate, the following http-headers must be added to the message:

- CertificateThumbprint - contains string value of thumbprint of certificate used for signing the message
- Client-Signature - contains the signature converted to Base64

payload for the signature will be concatenation of {httpMethod}|{UPPER(requestUrl)}|{SensorID}|{thumbprintValue}|{messagepayload} where the messagepayload is the JSON-body of the message to be sent.

For all V3-outgoing responses signature creation and -validation is done using the following procedure:

Verify message:

- CertificateThumbprint: <Platform certificate thumbprint>
- Server-Signature: *<Signature>* Base64EncodedByte string

payload for the signature will be concatenation of {statusCode}|{certificate.Thumbprint}|{messagepayload} where the messagepayload is the JSON-body of the message received.

3. Retrieve Actionlist V3

Retrieve Actionlist V3

Retrieve Action List

Pulls a hashed and salted list of Token IDs that are due for burn operations in the field for a given service; that the specific Terminal is hosting.

Use API Get: /V3/Sensors/Actionlist

With no message body.

Expected response

Field	Type	Description	M
data	SensorActionListResponse	object	Y
Success	Bool	True = Success	Y
Message	String	Optional extra details regarding results	N

SensorActionListResponse

Field	Type	Description	M
Collection	Dictionary (of String, List of String)	Hashed dictionary of Tokens with a list of actions for each token)	Y

Action List for Burn

This Burn List only serves to identify which tokens are due for a burn operation.

The actual burn activity is dependent upon the Token ID type, the necessity of a local SAM or transparent NFC, and the idiosyncrasies of a given Token ID's primitive onboard data types and the use thereof to achieve a given state on a card.

4. Retrieve Blacklist V3

Retrieve Blacklist V3

Retrieve Black List

Pulls a hashed and salted list of ETC Token IDs that are black listed for a given service; that the specific Sensor is hosting.

Use API Get: /V3/Sensors/Blacklist/{serviceId}

where serviceId is the serviceId provided by Local EcoSpace

With no message body.

Expected response

Field	Type	Description
Collection	String[]	Hashed Black List
Success	Bool	
Message	String	

5. Retrieve Whitelist V3

Retrieve Whitelist V3

Retrieve White List

Pulls a hashed and salted list of 42TECH Token IDs that are whitelisted for a given service; that the specific Terminal is hosting.

Use API Get: /V3/Sensors/Whitelist/{serviceId}

serviceId of type Long, id provided by local EcoSpace

With no message body.

Expected response

Field	Type	Description	M
data	String[]	Hashed Whitelist	Y
success	Bool		Y
message	String		Y

The array in the response consists of an ordered list of tokens that are whitelisted for the specified service.

If the requesting sensor has no certificate configured, the tokenvalues in the array are only hashed (SHA256 hash) and base64-encoded.

if the requesting sensor has a certificate configured, the tokenvalues in the array are concatenated with the thumbprint of the certificate, after which that result is hashed (SHA256) and base64-encoded.

If the sensorconfigurationsettings for the requesting sensor contains a setting 'SKIPWHITELISTHASH' the tokens are not hashed and not concatenated with the thumbprint of the certificate of the sensor (if available) and sent back 'in the clear'.

6. Send Heartbeat V3

Send Heartbeat V3

Send heartbeat

The Heartbeat is used to monitor EcoSpace Sensors and to update security and configuration data on the fly. It is a compound message consisting of multiple data objects and the heartbeat message must be signed.

Use API Post: /V3/Sensors/Heartbeat

With Message Body containing the following fields:

Message Body for API Post

Field	Type	Description	M
SensorLocation	GeoLocation	Data Object: see Geolocation	N
SensorConfigurationId	Long	Current Sensor Configuration ID provided by Local EcoSpace	Y
PreviousHeartbeatRTT	Long	Previous heartbeat round trip time	N
SensorId	Guid	Sensor ID provided by Local EcoSpace	Y

Data Object Geolocation

Field	Type	Description	M
Latitude	double	DD.dddddd° notation	N
Longitude	double	DD.dddddd° notation	N
Altitude	double	plus or minus sea level (in meters)	N
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
LocationAreaCode	int	Location area code (LAC) which is a 16 bit number thereby allowing 65536 location areas within one GSM PLMN.	N

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
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

Expected response

Field	Type	Description
Data	SensorConfiguration	Data Object see: 'SensorConfiguration' If no configuration update is available, the value is null
Success	Bool	
Message	String	

Data Object: SensorConfiguration

Field	Type	Description
SensorConfigurationId	Long	Current Sensor Configuration ID provided by Local EcoSpace
InsertDateTime	DateTime	Date & time of sensor configuration change.
PropertyBag	Dictionary of strings	Container with key value-pairs containing the configuration items for the particular sensor. E.g. Endpoint, Heartbeat Frequency, Service ID Active.

Heartbeat Response: SensorConfiguration Propertybag-example

Containing the configuration items for the particular sensor. E.g. Endpoint, Heartbeat Frequency, Service ID Active.

Key	Value (example)	Description
AMOUNT	0	Sets 'set price' in Sensor = Front end POS/
CERTIFICATEENABLED	TRUE	Signing enabled
CURRENCYCODE	EUR	Currency of service offered
DISPLAYMESSAGE	BUS 4G, LINE 84	Standard Idle Message
DISPLAYINACTIVEMESSAGE	Inactive	Standard Inactive Message
INCLUDEHASHEDTOKENID	TRUE	Used in special isntances where related services need to know if the same Token ID was used recently, e.g. Loyalty based cross service usage
ISACTIVE	TRUE	XAN command to activate terminal
SENDHEARTBEATEVERYXMILISECONDS	5000	Set frequency of heartbeat
SENSORAPIENDPOINT	https://oti-testnlhub.cloudapp.net/Sensor	Used to scale to multiple API endpoints in response to load
SERVICECAPABILITYTYPE	online	Sets basic mode of operation: Online Only Offline – Store & forward Offline –Reconcile Now

All values are Strings.

Design Note: The Property Bag is intended to be used to configure sensors on the fly; reference examples are available upon request.

7. Trigger Commit V3

Trigger Commit V3

Send TriggerCommit

After a trigger message is sent with requestmode OnlineAndCommit (requestmode 8) a triggercommit is expected by the hub. Only when the commit has been received by the hub, the triggermessage transaction will get the status 'Completed'.

Use API Post: /V3/Trigger/Commit

With Message Body containing the following fields:

Message Body for API Post

Field	Type	Description	M
Transaction	Transaction	Data object See: Transaction	Y

Data Object: Transaction

Field	Type	Description	M
Timestamp	String	Local timestamp format: yyyyMMddHHmmssfff concatenated with the time zone. Format [+/-]HHmm Example: 20151210191159000+0000	Y
Counter	Number	Counter of the sensor, incremented by one per transaction attempt	Y
SensorId	String	Sensor ID provided by Local EcoSpace	Y
ExternalTransactionId	String	Sensor reference number, Max Length 40 CHAR	N
ReferencedTransaction	String	Reference to a previous transaction	N
PropertyBag	Object	PropertyBag that must be included in case the decision for the current or previous transaction is Autonomous (offline verified transaction)	N

The PropertyBag shall be constructed as follows:

1). If the previous GST triggermessage was for an offline verified transaction, the PropertyBag shall contain the following:

TriggerMessage::Transaction.Propertybag[1]=(Key = “PreviousAutonomousTransactionID”, Value = PreviousTransactionID)

TriggerMessage::Transaction.Propertybag[2]=(Key = “PreviousAutonomousResult”, Value = PreviousAutonomousResult)

Where PreviousTransactionID shall be filled with the TransactionID of the previous offline verified transaction, and the PreviousAutonomousResult shall be filled with the value according to the table below.

2). If the current GST triggermessage is for an offline verified transaction, the PropertyBag shall contain the following:

TriggerMessage::Transaction.Propertybag[3]=(Key = “CurrentAutonomousResult”, *Value = CurrentAutonomousResult)*

Wherein the CurrentAutonomousResult shall be filled with the value according to the table below.

3). In case both the current as well as the previous triggermessage are for offline verified transactions, the PropertyBag shall be filled with all items above.

Field	Type	Description / Value
PreviousAutonomousResult or CurrentAutonomousResult	Number	0 = ok 2 = signature verification failed 3 = blacklisted 4 = Token expired 5 = Token issuer not supported 6 = Token status denied

Response

Response to a TriggerCommit is a HTTPStatusCode 204 (NoContent)

8. Trigger Request V3

TriggerRequest V3

Send Trigger Message

Trigger Pair Token ID / Terminal

After a transaction with a GST and optionally after offline risk management, the STAS sends a Trigger Message to the Hub.

The Trigger Message is constructed by the STAS, using the general construct of the Trigger Message and any extra items that may be required by the service (these are specified in the service's property bag) the Trigger Message is attempting to provision.

Once a trigger message is received, the Hub then routes it to the correct service or services for further processing.

Use API Post: /V3/Trigger

With Message Body containing the following fields:

Message Body for API Post

Field	Type	Description	M
Transaction	Transaction	Data object See: Transaction	Y
Tokens	Array of BaseTokens	Data Object see: BaseToken	Y
Sensor	BaseSensor	Data Object see: BaseSensor	Y
Service	BaseService	Data Object see: BaseService	Y
ServiceRequestData	ServiceRequestData	Data Object see: ServiceRequestData	Y

Data Object: Transaction

Field	Type	Description	M
Timestamp	String	Local timestamp format: yyyyMMddHHmmssfff concatenated with the time zone. Format [+/-]HHmm Example: 20151210191159000+0000	Y
Counter	Number	Counter of the sensor, incremented by one per transaction attempt	Y
SensorId	String	Sensor ID provided by Local EcoSpace	Y
ExternalTransactionId	String	Sensor reference number, Max Length 40 CHAR	N
ReferencedTransaction	String	Reference to a previous transaction	N
PropertyBag	Object	PropertyBag that must be included in case the decision for the current or previous transaction is Autonomous (offline verified transaction)	N

The PropertyBag shall be constructed as follows:

1). If the previous GST triggermessage was for an offline verified transaction, the PropertyBag shall contain the following:

TriggerMessage::Transaction.Propertybag[1]=(Key = "PreviousAutonomousTransactionID", Value = PreviousTransactionID)

TriggerMessage::Transaction.Propertybag[2]=(Key = “PreviousAutonomousResult”, Value = PreviousAutonomousResult)

Where PreviousTransactionID shall be filled with the TransactionID of the previous offline verified transaction, and the PreviousAutonomousResult shall be filled with the value according to the table below.

2). If the current GST triggermessage is for an offline verified transaction, the PropertyBag shall contain the following:

TriggerMessage::Transaction.Propertybag[3]=(Key = “CurrentAutonomousResult”,

Value = CurrentAutonomousResult)

Wherein the CurrentAutonomousResult shall be filled with the value according to the table below.

3). In case both the current as well as the previous triggermessage are for offline verified transactions, the PropertyBag shall be filled with all items above.

Field	Type	Description / Value
PreviousAutonomousResult or CurrentAutonomousResult	Number	0 = ok 2 = signature verification failed 3 = blacklisted 4 = Token expired 5 = Token issuer not supported 6 = Token status denied

If present, the key and value for the item CurrentAutonomousResult shall NOT be included in the construction of the HTD.

Data Object: BaseSensor

Field	Type	Description	M
Identifiers	Array	Data Object see: SensorIdentifier	Y
SensorLocation	Object	Data Object see: Geolocation	N

Data Object: SensorIdentifier

Field	Type	Description	M
IdentifierType	String	example.: IMEI / SNR / MAC / . . Max Length: 50	Y
IdentifierValue	String	e.g. serial number, ID, Max Length 255	Y

Data Object: BaseToken

Field	Type	Description	M
TokenType	String	Type of token. Here "GST21"	Y
TokenValue	String	Unique to abovementioned Token Type e.g. serial number, Token ID, Max Length 255	Y
Propertybag	Object	Data Object see: Propertybag	Y

Data Object: Propertybag, collection of keyvaluepairs.

Field	Type	Description	M
Key	String	Name of the key e.g. TSI, TransactionReceipt, HTD	Y
Value	String	Value corresponding to the key	Y

Data Object: BaseService

Field	Type	Description	M
ServiceId	Number	Service ID provided by Local EcoSpace	Y

Data Object: ServiceRequestData

Field	Type	Description	
RequestExternalIpAddress	String	External IP-Address of sensor device Example IPv4: 2.2.2.2	

RequestInternalIpAddress	String	Internal IP-Address of sensor device Example IPv4: 2.2.2.2	
RequestSensorLocalTimestamp	String	Local timestamp format: yyyyMMddHHmmssfff concatenated with the timezone. Format [+/-]HHmm Example: 20151210191159 000+0000	
Amount	Number	In cents, default 0	
CurrencyCode	String	ISO 4217 standard, 3 CHAR	
RequestMode	Number	1 = Online 2 = StoreAndForward 4 = Offline 8 = OnlineAndCommit	
PropertyBag	Object	Data Object see: PropertyB ag	

Return messagetype: ResponseMessage for RequestMode1

Field	Type	Description
Data	TransactionResponseBody	Identical structure and values to the request.
Message	String	Optional message
Success	Bool	For values see: Response Messages

TransactionResponseBody

Field	Type	Description
Transaction	Transaction	Identical structure and values to the request.
ResponseValue	int	
Propertybag	Object	Data Object see: Propertybag

Response Values

Message	ResponseValue	Description

SUCCESS	0	TriggerRequest was successfully handled by Ecohub.
UNKNOWN SENSOR	-2	Sensor used to send triggermessage is not an active Sensor in the Ecohub.
TRANSACTIONRECEIPT CHECK FAILED	-3	Validation of TMAC of triggerrequest failed.
UNKNOWN TOKEN FOR SERVICE	-4	There is no subscription for the requested service.
UNKNOWN SERVICE	-5	Requested service is not registered
REQUESTMODE <x> NOT SUPPORTED FOR REQUESTED SERVICE	-6	The requestmode (online, offline or storeAndForward) is not supported for requested service
TOKENTYPE NOT ALLOWED FOR SERVICE	-7	Requested service does not support tokentype
TOKEN IS NOT REGISTERED	-8	Unknown token used. Tokens need to be registered at hub before they can be used.
NO SERVICE-ENDPOINT FOUND	-9	No endpoint available in cache for generic servicehandler
<General Error>	-1	Message contains the error.

This list is not exhaustive. Other negative values may be used as well and indicate that the transaction has been declined.

Error codes

If a Sensor API call fails, the API returns an error code 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.

Propertybags Explained

Propertybags are used to introduce flexibility into the generic messaging design.

Depending on the Use Case and the Parent component E.G. Token, Service & Heartbeat response, Sensor Trigger Message response extra information can be included for the processing of a transaction that pertains to a specific implementation's idiosyncrasies.

For example:

A Token ID may employ a Token ID specific TMAC and seeds to prove card presence, these data can be included in the BaseToken property bag, for specialized authentication by a hub side bolt on module.

A Service Provider may desire more information than is typically present in a standard Trigger Message's Service specific field elements for calculation and provision of their service, these data can be included in the ServiceRequestData Propertybag.

A XAN Heartbeat response to a Sensor's heartbeat may require specific extra configuration commands related to a particular EcoSpace's terminal mgmnt needs, these data can be included in the in the Heartbeat response Propertybag.

A Sensor Trigger Message response may require a Picture or a balance be displayed, these data can be included in the Trigger message response Propertybag.

Examples of Propertybags

Token: BaseToken Propertybag-example

Used for additional Data related to Token e.g. TMAC, Engraved ID

Token Propertybag for GST 1.0-example

Key	Value	Description
TSI	0000000C00001300FF01000001000010	Containing ISIN_HOST and HOST-Counter
TOKENRECEIPT	0001E876D336660E55C7	TMAC of the GST-card
GSTVERSIONNR	0100	Last 2 Bytes of AID

TokenPropertybag for GST 2.1 example

Key	**Value**	**Description**
TSI	0A6FF2030000270000000000000000101000001000010	Containing ISIN_HOST and HOST-Counter
TOKENRECEIPT	11012CF85020CCAF5432	TMAC of the GST-card
GSTVERSIONNR	0210	Last 2 Bytes of AID
HTD	6F4AFE9BC638A766CB70CB957C0EAA58C55A8B973C11058B3A8988672D6E128C	Hash of transactionvalues

Token Propertybag for MFA-example

Key	**Value**	**Description**
ChipId	F31EB8EA	UID of the card (outsideId)
TMV	0001E876D336660E55C7	TMAC of the MFA-card
TMC	010000	Card Transaction counter

CredentialFile		128 bytes (64 bytes data and 64 bytes ECC signature)
EncPreviousSensorId		Encrypted value of id of sensor where previous transaction took place

Service: ServiceRequestData Propertybag-example

Contains specific data for a service. This could be really anything, depending what extra data is required for a service that is not available in the general fields of the triggermessage.

Key	Value	Description
UniqueTokenIdentification	Jf2wBETh2EstFHZBrgasjQpCPL7vLGGWwkowVQNQuZs=	Hashed value of used token

Sensor Trigger Message response: Responsemessage Propertybag-example

The propertybag of the trigger message response contains data to be displayed or processed at the sensor. The table below contains a few examples of possible data that can be present in de trigger message response propertybag.

Key	Value	Description
Picture		Bytearray containing data of the picture linked to the token
Wallet	Ecowallet	Type of wallet that is returning a balance
Balance	2350	Balance in cents
Currency	EUR	Currency of the wallet

9. Trigger Rollback V3

Trigger Rollback V3

Send TriggerRollback

After a trigger message is sent with requestmode OnlineAndCommit (requestmode 8) a triggerrollback can be sent to the hub to rollback the transaction that is waiting for a commit.

Use API Post: /V3/Trigger/Rollback

With Message Body containing the following fields:

Message Body for API Post

Field	Type	Description	M
Transaction	Transaction	Data object See: Transaction	Y

Data Object: Transaction

Field	Type	Description	M
Timestamp	String	Local timestamp format: yyyyMMddHHmmssfff concatenated with the time zone. Format [+/-]HHmm Example: 20151210191159000+0000	Y
Counter	Number	Counter of the sensor, incremented by one per transaction attempt	Y
SensorId	String	Sensor ID provided by Local EcoSpace	Y
ExternalTransactionId	String	Sensor reference number, Max Length 40 CHAR	N
ReferencedTransaction	String	Reference to a previous transaction	N
PropertyBag	Object	PropertyBag that must be included in case the decision for the current or previous transaction is Autonomous (offline verified transaction)	N

The PropertyBag shall be constructed as follows:

1). If the previous GST triggermessage was for an offline verified transaction, the PropertyBag shall contain the following:

TriggerMessage::Transaction.Propertybag[1]=(Key = “PreviousAutonomousTransactionID”, Value = PreviousTransactionID)

TriggerMessage::Transaction.Propertybag[2]=(Key = “PreviousAutonomousResult”, Value = PreviousAutonomousResult)

Where PreviousTransactionID shall be filled with the TransactionID of the previous offline verified transaction, and the PreviousAutonomousResult shall be filled with the value according to the table below.

2). If the current GST triggermessage is for an offline verified transaction, the PropertyBag shall contain the following:

TriggerMessage::Transaction.Propertybag[3]=(Key = “CurrentAutonomousResult”, *Value = CurrentAutonomousResult)*

Wherein the CurrentAutonomousResult shall be filled with the value according to the table below.

3). In case both the current as well as the previous triggermessage are for offline verified transactions, the PropertyBag shall be filled with all items above.

Field	Type	Description / Value
PreviousAutonomousResult or CurrentAutonomousResult	Number	0 = ok 2 = signature verification failed 3 = blacklisted 4 = Token expired 5 = Token issuer not supported 6 = Token status denied

Response

Response to a TriggerCommit is a HTTPStatusCode 204 (NoContent)