Dr. Oliver Nagy

Implementation of the Kapsch Open EFC Interface into the CZ Pilot System
Definition of used Terms and Topologies for the CZ Pilot System (1)

• **Client**
  The Proxy, including all device management functionality, and the respective OBUs are summarized as the Client or the “OBU Front End” (according to PT20).

• **Context Layer**
  The Context Layer provides the interface of the data flow from the OBU to the Central System or vice versa in a defined language spoken between OBU Client context servers and Central System context server. The Context Layer does NOT represent OBU specific (proprietary) data; it is a generic instruction set, enabling communication between Central System and OBUs.
  
  The Context Layer supports any kind of implementation of the Client (OBU+Proxy) independently of the functional split between OBU and Proxy (Thin OBU Clients (like Kapsch Area OBUs) or Thick OBU Clients).

• **Context Server**
  The Context Server implements the translation between the proprietary OBU Client language (protocol) through the Context Layer instruction set.
  
  Implementation of Context Server on OBU Client side is responsibility of the OBU Client-supplier.
  KTC is responsible for implementation of the Context Server on Central System side.
Definition of used Terms and Topologies for the CZ Pilot System (2)

- **Arbitration Layer**
  The AL provides the dispatching functionality in order to handle all Clients (=OBUs proxy).
  It represents multiple data connection between CS and OBU Clients and so, provides basic interoperable (EETS) OBU handling functionality (=handling of interoperable Clients as interface to the central system Access Layer. 

  The Arbitration Layer is a passive architecture, the links between singular or multiple OBUs and dedicated functions of the Central System are given by the WSDL (Web Service Definition Language) file. It is implemented as a SOA (Service Oriented Architecture) by a Webservice.

- **Telematic Message Interface**
  The Telematic Message Interface enables a direct connection for Telematic Messages between a Central System-independent Telematic Applications and the OBU through the open interface.
Scope of PT20 for future Application Interface Definition for CN/GNSS Toll Systems (ISO 17575)

CEN TC278 PT20 scope ISO TS 17575 data definitions

Service Provider Domain

Toll Context Description (Toll Objects, Tariff Scheme, Tariff Tables, …)

Supporting Back Office Functions (Proxy, Map Matching, OBE-Communication, …)

Use Data

Back Office

OBE

Front-End

Thick Client

Thin Client
Implementation of Arbitration Layer & Context Layer

Central System (Toll Charger & Service Provider)

Arbitration Layer (Dispatcher)

Access Layer

Client Supplier 2

DSRC interface will not be discussed!!!
Interoperability with GNSS/CN Service Provider: Scope of CZ Pilot
General Requirements

➢ The central system defines the roads to be charged and provides them in the form of geo-objects (Charge Objects). The OBU Client has to integrate these Geo-Objects into its system, whether it works with a matching functionality or dedicated toll gates. This ensures a decoupling of the OBU Client system and the central system in terms of Geo Information. No specific Geo Data Base has to be shared.

➢ The source for the Geo objects is maintained and updated correctly within the central system.

➢ Geo Objects are defined according to parts of ISO 14825.
  ➢ (ISO14825:2004; Intelligent Transportation Systems – Geographic Data Files (GDF))

➢ The Proxy must get and work with Geo Objects from the central system, but must be responsible for their own internal Geo information system (GIS), e.g. digital map, and the integration of the Geo objects into this proprietary system.
General OBU Client Requirements

- The Client must be able to buffer tolling transactions for a defined number of days of operation for each OBU in case of incomplete or incorrect transmission (e.g. outside wireless service coverage, WAN problems) between OBU and Proxy or between Proxy and Central System until successful transmission to the Central System.

- The Client must provide a certain percentage of all transactions within a certain time as predefined by CS/ System Operator.

- The Proxy must have capability to handle at least 5000 OBUs simultaneously (this figure is project specific) for both, air and WAN interface side.

- The Client must execute transactions according to priorities, toll transactions always with highest priority and telematic transactions according to defined priority.
The tolling data has priority against all other data (telematic messages, other services).

The limited bandwidth and capacity of the air interface and within the proxy as well has to be reserved by 100 % for tolling data. In case of free capacity of the dedicated channel this can be used for other services (e.g. Telematic Messages, OBU specific device control, …), separately prioritized.

The Client must assign the full data transmission capacity (e.g. data rate) and performance (e.g. delay times) of wireless service (e.g. CN GSM-/GPRS-service) to the transmission of tolling data between Proxy and OBUs at any time.

The Client must assign the highest priority (against Telematic Messages) to Tolling Messages.

The Client must transmit all tolling transactions immediately after correct processing. The maximum transmission delay is predefined by Central System Operator.

The Client should have capability to store at least 100 Telematic Transactions per OBU.

The Client shall have capability to store at least 10 Telematic Transactions per OBU.
General Interface Implementation

- Security
- Authentication
- Privacy

According to VPN principles
Tolling Messages (1/5)

- **GetTariffList**
  
  This function (Webservice) is executed by the Client Proxy to get the whole tariff list or to get only newer entries.

  **TariffSerialNumber**
  
  If specified, indicates that the proxy is only interested in records with a serial number higher than the one specified. 0 = from the first record.

  **MaxCount**
  
  If <> 0 the Webservice returns only the first MaxCount tariffs.
The method returns an array of objects of type TariffEntry. A TariffEntry contains the following attributes:

- only new changed items – serial number defines last item

Tariff entry serial number
TollSectionID.
Axle class.
Weight class
Emission class
From transaction time in UTC, in XML string format (HH:mm:ss).
To transaction time UTC, in XML string format (HH:mm:ss).
Valid from date in UTC in XML string format (yyyy-MM-ddTHH:mm:ss). The tariff is valid only between the VFD and the VTD.
Valid to date in UTC in XML string format (yyyy-MM-ddTHH:mm:ss). Can be NULL. The tariff is valid only between the VFD and the VTD.
Fee
TollSection tariff version.
Tolling Messages (2/5)

- **GetTollSections**
  
  This function (Webservice) is executed by the Client Proxy to get all defined toll sections or to get only newer entries.

  **TSSerialNumber**  
  If specified, indicates that the proxy is only interested in records with a serial number higher than the one specified. 0 = from the first record.

  **MaxCount**  
  If <> 0 the Webservice returns only the first MaxCount tariffs.
The method returns an array of objects of type TollSection. A TollSection contains the following attributes:

- **TSSerialNumber** TollSection serial number.
- **TSID** Toll section ID.
- **TSTID** Toll station ID
- **TSN** Toll section name
- **TSTN** Toll station name
- **FGPSLA** From GPS (toll section start position) latitude in decimal degrees (xxx,xxxxxx)
- **FGPSLO** From GPS (toll section start position) longitude in decimal degrees (xxx,xxxxxx)
- **FN** From name. For example name of the street where the Toll section starts.
- **FKM** Start km
- **TGPSLA** To GPS (toll section end position) latitude in decimal degrees (xxx,xxxxxx)
- **TGPSLO** To GPS (toll section end position) longitude in decimal degrees (xxx,xxxxxx)
- **TN** To name. For example name of the street where the Toll section ends.
- **JN** All junctions which are related to the toll section (start junction, end junction, junction between start and end)
- **TSL** Toll section length in meters
- **AVGT** Average time to pass the toll section in seconds
- **TST** Toll section type. (0…section, 1…position)
- **TSTT** Toll station type. (SKE, MAS, VG, Infrared, Tollplaza)
The method returns an array of objects of type TollSection. A TollSection contains the following attributes:

**CONTINUATION:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSGPSLA</td>
<td>Toll station GPS latitude in decimal degrees (xxx,xxxxxx)</td>
</tr>
<tr>
<td>TSGPSLO</td>
<td>Toll station GPS longitude in decimal degrees (xxx,xxxxxx)</td>
</tr>
<tr>
<td>RN</td>
<td>Route number</td>
</tr>
<tr>
<td>FRC</td>
<td>Functional road class</td>
</tr>
<tr>
<td>AC</td>
<td>Area code</td>
</tr>
<tr>
<td>SO</td>
<td>Scheme owner</td>
</tr>
<tr>
<td>VFD</td>
<td>Valid from date in UTC in XML string format (yyyy-MM-ddTHH:mm:ss). The toll section is valid only between the VFD and the VTD.</td>
</tr>
<tr>
<td>VTD</td>
<td>Valid to date in UTC in XML string format (yyyy-MM-ddTHH:mm:ss). The toll section is valid only between the VFD and the VTD.</td>
</tr>
<tr>
<td>TSV</td>
<td>Toll section version</td>
</tr>
</tbody>
</table>
• **GetOBUState**

  This function (Webservice) is executed by the Client Proxy to get the whole status information for all onboard units which are related to one or more ContractProviderIDs.

  **ContractProviders** Contains an array of Contract Provider that should be processed.

  Each ContractProvider contains the following structure:

  **ContractProvider** Contract provider ID associated to the TAG.

  The method returns an array of objects of type OBUState. A OBUState contains the following attributes:

  **ContractProvider** Contract provider ID associated to the TAG.
  **ManufacturerID** ManufacturerID associated to the TAG.
  **OBUID** Serial number associated to the TAG.
  **OBUState** OBU state information
Tolling Messages (4/5)

• **InsertTransactions**

This function (Webservice) is executed by the Client Proxy to insert one or more transactions into the Commercial Back Office. The function accepts the following parameters:

- **Transactions (IN)** Contains an array of toll transaction packages that should be processed.
- **Result (OUT)**
  - Zero = Ok
  - Other values = Error code
- **ErrorMessage (OUT)** Description of the error, in case Result is different of zero.

Each TollTransactionPackage contains the following structure:

- List of Tolling Vehicle (OBU metadata)
- List of (to Tolling Vehicle) corresponding transactions
Tolling Messages (5/5)

- InsertOBUEvents

  This function (Webservice) is executed by the Client Proxy to set OBU events within the Commercial Back office.

  **Events (IN)**
  Array of Event objects (see definition below)

  **Result (OUT)**
  0 = Ok
  Other values = Error code

  **ErrorMessage (OUT)**
  Description of the error, in case Result != 0.

The Event object has the following attributes:

- **ContractProvider**
  Contract provider associated to the OBU

- **ManufacturerID**
  OBU manufacturer ID

- **OBUID**
  unique identifier of the OBU

- **DateTime**
  UTC event time, in XML string format (yyyy-MM-ddTHH:mm:ss).

- **EventData**
  Structure which contains the event data
The EventData has the following structure:

- EventId: OBU event ID
- Severity: Severity
- Parameter_A: Parameter A
- Parameter_B: Parameter A
- Parameter_C: Parameter A

**Tolling Messages (5a/5)**
Telematic Message Interface Topology
Telematic Messages

Telematic Messages are a bidirectional communication path, in the implementation these functionality is represented by two SOAs (Service Oriented Architectures), realized by Web Service. This leads to two WSDL (Web Service Definition Language) files, one within the Client and one on the Central System side in order to allow the bidirectional communication.

- SetTelematicMessage(Priority, Type, Address[], Object)
- GetTelematicMessage(Priority, Type, Address[])
- SetTelematicOBUCfg(Priority, Address[], Object)
- GetTelematicOBUCfg(Priority, Address[], Object)
**Enforcement Messages**

- **(Attribute) Data Specification**

  The attributes have to be supported by any (3rd party) OBU.

  This includes all enforcement relevant attributes defined by the system operator as well as attributes defined through the draft [CCC] standard.

  [CCC] prCEN/TS 0000:2008 (Draft)

  Road Transport and Traffic Telematics (RTTT) – Electronic fee collection systems (EFC) – Compliance check communication for autonomous systems

  Within the Pilot System Enforcement is only realized by the DSRC-Interface, no other enforcement is within the scope of the Pilot!
CZ Pilot System Properties

Since KTC is delivering a Pilot GNSS system in CZ Republic, it is
• a good opportunity to bring in such an interface for testing & evaluation purposes:
• even if it will be implemented before the standard is released and definitions are not synchronized, it
  will at least be very close and show the principle ….

We talk about the tolling functionality, including toll objects and OBU stati, but leaving out
• OBU maintenance issues and interfaces needed for contractual and operational purposes between
  Client and Central System
→ the specification which will be delivered to RSD end of month …

The interface principle is that the Central System predefines exactly:
1.) The communication frame work (Arbitration Layer)
2.) The communication messages (Context Layer)
3.) The Tolling parameters (e.g. tariff list)
4.) How the client has to return toll objects and its contents
   (Charge report from the client and a charge response to the client)
Further Details

All further details will be released by the Pilot Kapsch Open EFC Interface Specification, V1.0, July 2008, owned by RSD.
Thank you!