Cello-CANiQ

Integration Manual

Proprietary and Confidential

Copyright © 2014 Pointer Telocation

Version 1.1

Revised and Updated: March 28, 2017

Copyright © 2017 by Pointer Telocation, Ltd.

Legal Notices

IMPORTANT

- All legal terms and safety and operating instructions should be read thoroughly before the product accompanying this document is installed and operated.
- This document should be retained for future reference.
- Attachments, accessories or peripheral devices not supplied or recommended in writing by Pointer Telocation Ltd. May be hazardous and/or may cause damage to the product and should not, in any circumstances, be used or combined with the product.

General

The product accompanying this document is not designated for and should not be used in life support appliances, devices, machines or other systems of any sort where any malfunction of the product can reasonably be expected to result in injury or death. Customers of Pointer Telocation Ltd. Using, integrating, and/or selling the product for use in such applications do so at their own risk and agree to fully indemnify Pointer Telocation Ltd. For any resulting loss or damages.

Warranty Exceptions and Disclaimers

Pointer Telocation Ltd. Shall bear no responsibility and shall have no obligation under the foregoing limited warranty for any damages resulting from normal wear and tear, the cost of obtaining substitute products, or any defect that is (i) discovered by purchaser during the warranty period but purchaser does not notify Pointer Telocation Ltd. Until after the end of the warranty period, (ii) caused by any accident, force majeure, misuse, abuse, handling or testing, improper installation or unauthorized repair or modification of the product, (iii) caused by use of any software not supplied by Pointer Telocation Ltd., or by use of the product other than in accordance with its documentation, or (iv) the result of electrostatic discharge, electrical surge, fire, flood or similar causes. Unless otherwise provided in a written agreement between the purchaser and Pointer Telocation Ltd., the purchaser shall be solely responsible for the proper configuration, testing and verification of the product prior to deployment in the field.

POINTER TELOCATION LTD.'S SOLE RESPONSIBILITY AND PURCHASER'S SOLE REMEDY UNDER THIS LIMITED WARRANTY SHALL BE TO REPAIR OR REPLACE THE PRODUCT HARDWARE, SOFTWARE OR SOFTWARE MEDIA (OR IF REPAIR OR REPLACEMENT IS NOT POSSIBLE, OBTAIN A REFUND OF THE PURCHASE PRICE) AS PROVIDED ABOVE. POINTER TELOCATION LTD. EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, SATISFACTORY PERFORMANCE AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL POINTER TELOCATION LTD. BE LIABLE FOR ANY INDIRECT, SPECIAL, EXEMPLARY, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOSS OR INTERRUPTION OF USE, DATA, REVENUES OR PROFITS) RESULTING FROM A BREACH OF THIS WARRANTY OR BASED ON ANY OTHER LEGAL THEORY, EVEN IF POINTER TELOCATION LTD. HAS BEEN ADVISED OF THE POSSIBILITY OR LIKELIHOOD OF SUCH DAMAGES.

Copyright © 2017 by Pointer Telocation, Ltd.

Intellectual Property

Copyright in and to this document is owned solely by Pointer Telocation Ltd. Nothing in this document shall be construed as granting you any license to any intellectual property rights subsisting in or related to the subject matter of this document including, without limitation, patents, patent applications, trademarks, copyrights or other intellectual property rights, all of which remain the sole property of Pointer Telocation Ltd. Subject to applicable copyright law, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), or for any purpose, without the express written permission of Pointer Telocation Ltd.

© Copyright 2017. All rights reserved.

Disclaimer

Cello-CANiQ is an open platform allowing the user to implement various CANBUS connectivity configurations with regards to the ECU parameters being captured or queried, as well as the querying rate. While using the vehicle's OBDII port, the Cello-CANiQ sends queries to the diagnostics ECU. In such installations, it is possible that unprofessional user defines configuration which results in errors on the OBD port. In other cases, the installer may choose to connect the device directly to the vehicle bus, via wired connection and not a dedicated connector – installation type which may be referred by a vehicle manufacturer as a cause for warranty remit . User shall use only validated installation and device configuration which were officially recommended by Cellocator.

In no event shall Pointer be liable for any direct, indirect, incidental, special, punitive or consequential damage, losses or expenses relating to or arising from the Cello-CANiQ, including without limitation, its installation, programming, configuration and use thereto; provided however, that Pointer warrants to the buyer that during the period of one year commencing on the date the Cello-CANiQ is purchased the Cello-CANiQ shall conform in all material respects with the performance specifications, attached to the unit purchased.

Copyright © 2017 by Pointer Telocation, Ltd.

Table of Contents

1 Introduction 5
1.1 Scope 5
1.2 Purpose 5
1.3 Target Audience 5
1.4 References 6
1.5 Revision history 6
1.6 Acronyms 6
2 General Description 7
2.1 Overview 7
2.2 Cello-CANiQ Block Diagram 7
2.3 Prerequisites 8
2.4 Network Architecture 8
2.5 Objectives of the Cello-CANiQ Server Integration 9
3 Cello-CANiQ Configuration Management 10
3.1 Parameters, Triggers, Events, and Data Flow 10
3.2 Typical Type 11 Uplink CAN Activity and Reporting 10
3.3 Cello-CANiQ Configuration Tool 12
3.3.1 PL and XML 12
3.3.2 XML Backend Management Practices 13
3.3.3 canconfig.xml File Structure – the Diagram.xml file 13
4 Server-Side Integration 17
4.1 Integration Modes 17
4.2 Generic Type 11 Handler Block Diagram 17
4.2.1 Typical Type 11 Scenarios 18
4.3 OTA Interface Benefits 19
4.3.1 Features of the New Modular Type 11 Protocol 19
4.4 Type 11 Protocol Structure 20
4.4.1 Message Components 20
4.4.2 Type 11 Module Structure 22
4.4.3 Summary of Supported Type 11 Modules 23
4.5 Automatic GNSS Antenna Status 24
5 Cellocator Integration Package 25

Copyright © 2017 by Pointer Telocation, Ltd.

1 Introduction

This manual gives instructions and technical information for integrating Cello-CANiQ units with operational fleet management SW platforms. This enables retrieving information from fleet vehicles via the CAN interface and integrating said information into the server-side applications.

1.1 Scope

This document is part of the Cellocator documentation set for the Cello-CANiQ. It is assumed the reader is familiar with the Cellocator products, communication protocols and device provisioning capabilities.

1.2 Purpose

This guide describes how the server developer can use the provided interface to perform the following:

- Retrieve information from fleet vehicles via the CAN interface and integrate said information into the server.
- Build a gateway protocol designed to interconnect with multiple Cello-CANiQ units.

1.3 Target Audience

The target audience is TSP's and SW integrators server-side developers adding Cello-CANiQ units to their existing fleet management system. This manual supports two types of software integrators depending on the level of integration required by the existing server application.

The first type of integration is intended to address fleet management systems working directly with Cellocator wireless protocols and thus self-implementing all the low level communication layers. Such integrators need to have good knowledge and understanding of the Type 11 modular protocol, its data entities and data flow.

The second type of integration is intended for TSPs and software integrators already using the Cellocator integration package or those who are new to the Cellocator APIs and OTA interfaces and considering their optional integration paths. The Cellocator Integration package is a set of backend gateway components designed to hide the internal message processing and parsing of the OTA-protocol from the integrator. It does this by transforming the integration task from a low level, communication and protocol management task into a database level, explicit and parsed information integration mission. Customers using the Cellocator Gateway benefit from a quicker and easier integration process and are entitled to software upgrades, technical support and more.

Copyright © 2017 by Pointer Telocation, Ltd.

1.4 References

No.	Document Name
1	Cellocator Wireless Communication Protocol This document includes the complete OTA protocol (Fleet, Type 11) and the Type 11 file structures
2	Cellocator Cello Programming Manual
3	Cellocator Serial Communication Protocol
4	Cellocator CSA Programming Manual
	Cellocator Programmer Manual (including the CAN Editor)
4	Evaluation Manual
5	Cellocator Integration Package [Full Edition] v2.0 Manual
6	Cello-CANiQ product Overview

1.5 Revision history

Version	Date	Description
1.0	Novenber 2014	Original version
1.1	March 2017	Update the path of the XML files

1.6 Acronyms

Abbreviation	Definition
ACK	Acknowledge
API	Application Programming Interface
CCC	Command and Control Center
CSA	Car Safety Application
OTA	Over the air
TSP	Telematic Service Provider

Copyright © 2017 by Pointer Telocation, Ltd.

2 General Description

2.1 Overview

Cello-CANiQ is a new member of the Cellocator product line and is designed to integrate CAN bus information from remote vehicles into the system server. The Cello-CANiQ uses the network infrastructure already implemented in Cello-IQ together with an extended fleet protocol to support an application layer carrying CAN bus events and data.

2.2 Cello-CANiQ Block Diagram

The following figure shows a block diagram of the Cello-CANiQ device.

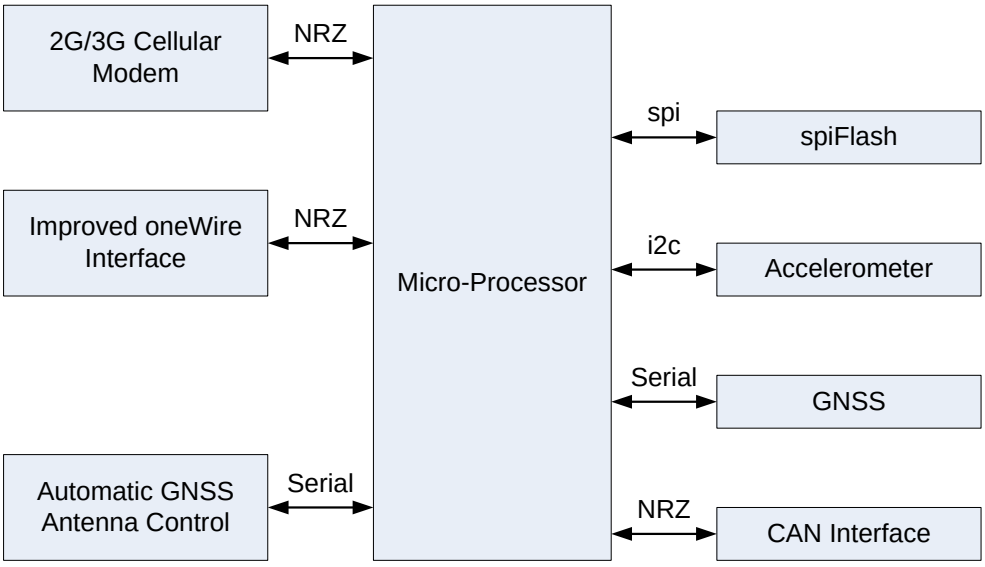


Figure 1 – Cello-CANiQ Block Diagram

- **2G/3G Cellular Modem** – Cellular modem enabling Internet connectivity and voice services.
- **Improved oneWire Interface** – Provides a current driver capable of powering up to 4 1-wire temperature sensors. The interface is also responsible for communicating with the sensors.
- **Automatic GNSS Antenna Control** – Provides hardware support for the external GNSS antenna, including fault detection (shorted or disconnected) and automatic internal/external antenna selection for optimal GNSS reception.
- **spiFlash** – Provides non-volatile storage for configuration files and event logs. Communicates with the system via the serial peripheral interface.
- **Accelerometer** – Measures the magnitude and direction of acceleration. This is a MEMS-based device, using an i2C serial interface to transfer movement data to the system which translates it into driver behavior data.
- **GNSS** -
  - Supports both GPS and Glonass
  - Supports new GNSS Chip-Set based on STM 8088
  - Enhanced GNSS performance

Copyright © 2017 by Pointer Telocation, Ltd.

- **CAN Interface -**
  - CAN bus interface supporting both OBD2(ISO 15765-4 / SAE J2284) and CAN2.0
  - Simple OBD2 harness used as a power source
  - User configurable filters, triggers and actions
  - Pre-configured FMS variables
  - Pre-configured OBD2 standard PIDs
  - Graphical user tool for CAN triggering

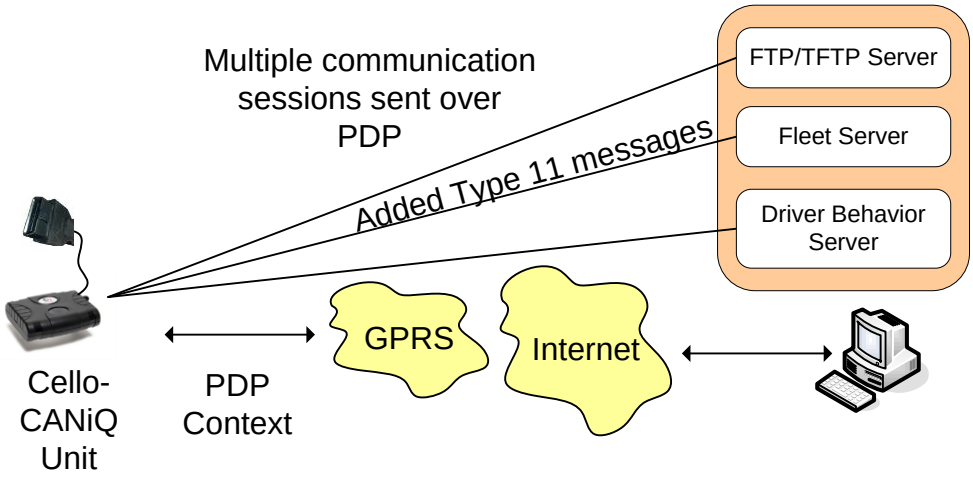
2.3 Prerequisites

Users should become familiar with the following documents:

- Cello-CANiQ product overview
- Cellocator Programming Manual
- Cellocator OTA wireless protocol
- CSA Programming Manual
- Cellocator Integration Package [Full Edition] Manual if working in this mode

It is assumed the user has already set up an evaluation environment and demonstrated communication interchange between the evaluated Cello-CANiQ unit and the communication center provided by Cellocator.

2.4 Network Architecture



Cello-CANiQ has extended the network connectivity capabilities of the Cello-IQ unit by adding support for Type 11 fleet messages. Type 11 messages are true modular messages with user-configurable structure. CAN related information elements are sent via Type 11 modules. The CAN information elements sent by the Cello-CANiQ result from a triggering mechanism configured by the user via an advanced graphical editor available in the Cellocator Programmer SW utility.

Copyright © 2017 by Pointer Telocation, Ltd.

2.5 Objectives of the Cello-CANiQ Server Integration

Cello-CANiQ is ready for integration into any TSP software platform with minimal development and integration effort. This document facilitates the process of turning Cello-CANiQ from an in-vehicle physical product to a complete CAN solution.

The server-side integration process requires the server-side to receive and parse the incoming messages sent by the Cello-CANiQ units. If the server already supports Cello-IQ then the integration efforts will only focus on Type 11 server-side implementation.

The CAN related data sent by the Cello-CANiQ unit includes information designed to support server-side applications such as remote diagnostics, remote maintenance and remote vehicle-trouble management and fuel management applications . The information element sent by the units is configurable and enables users to include only information associated with their server-side applications.

The major objectives of the Cello-CANiQ Server Integration are as follows:

- Receiving and parsing Type 11 events originated by the CAN
- Sending Type 11 Commands to the unit.
- Processing CAN-originating events
- Processing antenna state changes

Copyright © 2017 by Pointer Telocation, Ltd.

3 Cello-CANiQ Configuration Management

Users can create customized message configurations to be broadcast by the remote units via Type 11 messages to the system server.

3.1 Parameters, Triggers, Events, and Data Flow

Cello-CANiQ supports CAN connectivity designed to widen the visibility of the server side applications to the vehicle information sources. It supports both CAN and OBD2 vehicle busses.

The flow of CAN data can be represented as follows:

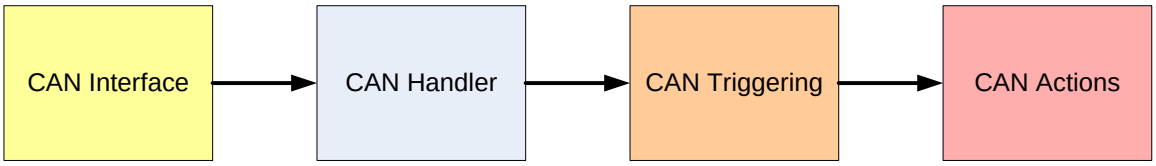


Figure 2 – CAN Data Flow

- **CAN Interface** – The physical and data link layers handle wire level and framer level messages.
- **CAN Handler** - OBD2 Query/Response Manager, CAN Message interceptor. Parses the raw data in the CAN application messages and maps specific sequences of bits to Cello-CANiQ parameters.
- **CAN Triggering** – Generates events based on the changing values of the CAN parameters. For example, if the motor RPM is greater than 5,000 for a period exceeding one minute, an action is triggered.
- **CAN Actions** – As a result of true trigger criteria, the system generates configurable Type 11 messages and/or GPIO output patterns. For example, a message can be sent containing all the information relevant to the state that triggered the message, or a waveform can be generated on a GPIO output (for example, a waveform that causes a buzzer to sound a number of beeps).

3.2 Typical Type 11 Uplink CAN Activity and Reporting

When CAN bus events are detected by the Cello-CANiQ device, the device processes the new information and broadcasts messages to the system server. The following diagrams describe how Type 11 messages resulting from Cello-CANiQ triggers are built.

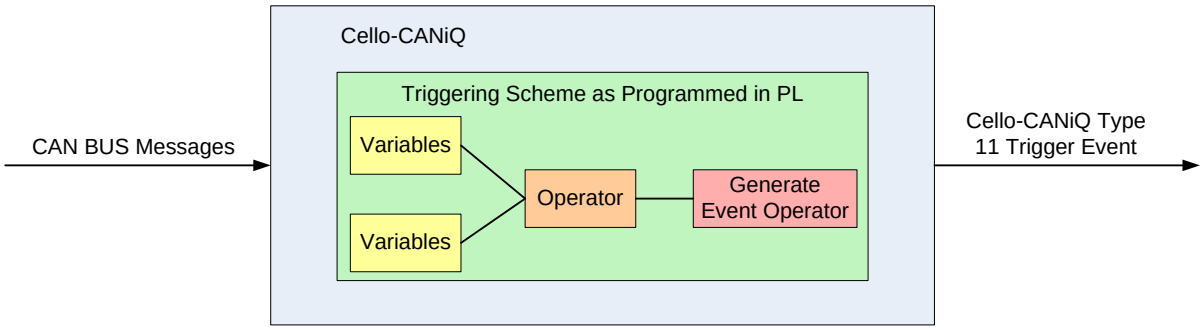


Figure 3 – CAN Activity and Reporting

Copyright © 2017 by Pointer Telocation, Ltd.

The CAN Editor module, which resides within the Cellocator programmer tool, illustrates how variables are included in Type 11 messages. The message in the example in Figure 4 is composed of the following mandatory Type 11 modules:

- 6 and 7 – Contain the GPS date, location, and time stamp.
- 25 – Identifies the generating event operator, in this case, the green operator with ID 1E00.

The following module has also been added:

- 2 – Contains values of the selected variables, in this case, the blue variable with ID 4004 which represents the engine RPM.

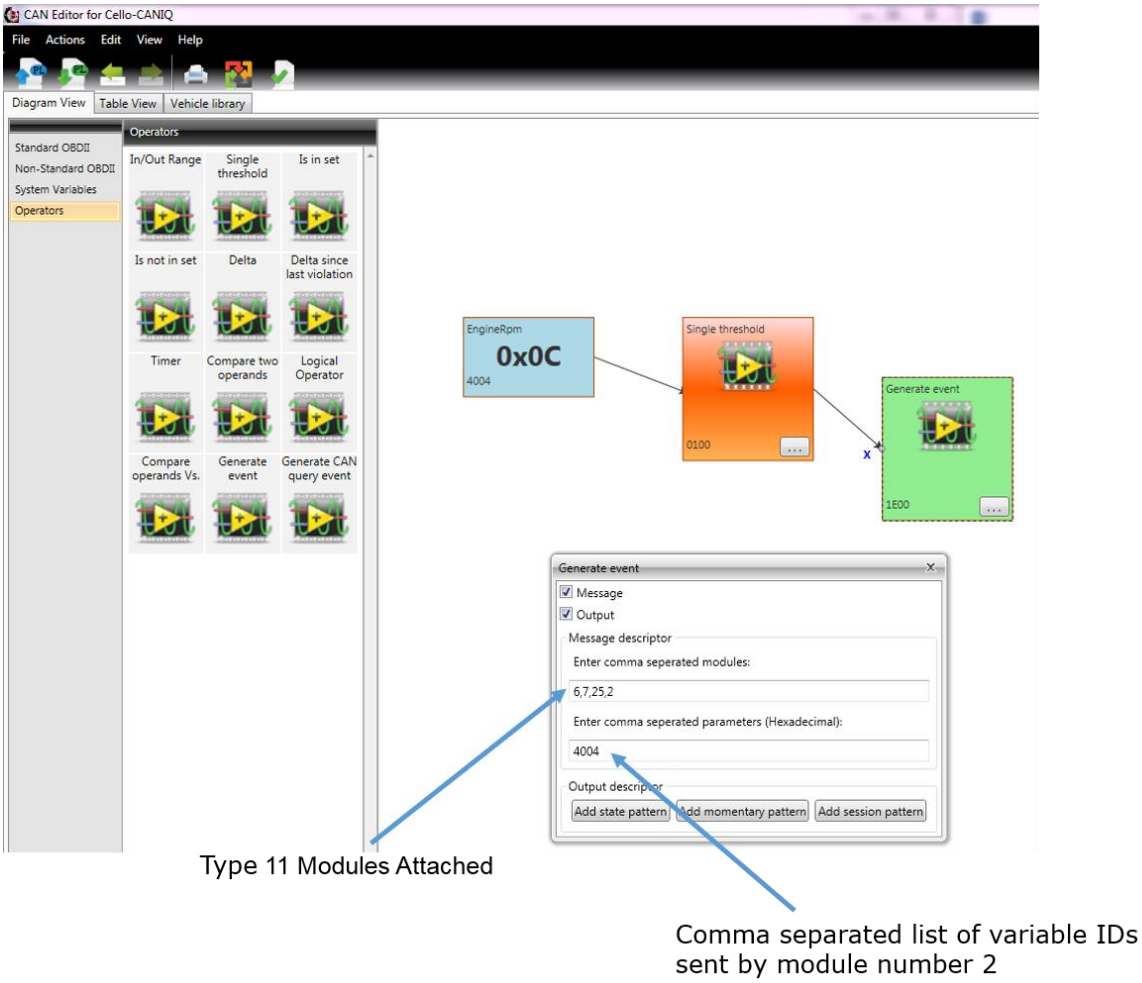


Figure 4 – CAN Editor

Typical parsing results are shown below:

- For detailed description of Type 11 structure please referrer to Type 11 Protocol Structure.
- The following is an example of how a Type 11 message is parsed:

4D4347500B53B60100480146007E0000000000	Header
080600000121011400	FW ID

Copyright © 2017 by Pointer Telocation, Ltd.

0613000404020D5545A30310145703EC2C0000290000	GPS Location Stamp
070700012C080B150A0D	GPS Time Stamp
1905001E00010000	Trigger Event ID
0246001E0056C1F71909804004462C00008240047C0000008	
5400400000000834004000000008140045EBD320184400464	
270000864004000000008740040000000	Variables
088400400000000	Dump List

- Typical CAN Type 11 Variable Dump List (module number 2) Parsing

02	Module ID
4600	Module Length
1E00	Operator ID
56C1F719	PL Signature
01	Number of Variables
0440	First Variable ID
04	Variable length (4 Bytes)
462C0000	Variable data: 0x00002c46

3.3 Cello-CANiQ Configuration Tool

The CAN Editor is the Cello-CANiQ configuration tool and it provides a graphical interface for configuring CAN-related information sources with user defined behavior. The tool runs on your PC and enables you to:

- Define parsing criteria of data sent by the remote units to the server
- Drag and drop parameters into the work arena
- Combine parameters into expressions using logical operators
- Output behavior: Generate the following types of event resulting from expressions:
  - Type 11 messages
  - GPIO patterns, for example, an electrical waveform to sound a buzzer

Typically, it is necessary to connect a Cello-CANiQ unit to the PC via a serial port. In this case, before downloading your new configuration to the unit, the software verifies that your logic is valid or prompts you to make corrections. Even if no unit is connected, the software is able to detect syntax errors and prompt you to correct them.

When your configuration is complete, you can download it in a single operation to all the units in your system *over-the-air* using the Cellocator+ or equivalent tools.

3.3.1 PL and XML

The Cellocator Programmer is used to program the Cello-CANiQ device. However, programming of the device's CAN parameters is performed with the CAN Editor, an add-on application which can be accessed via the Cellocator Programmer interface. Both the Cellocator Programmer and the CAN Editor save information in the same ***.pl** file which can be downloaded to a remote Cello-CANiQ units or a device connected via serial connection.

Copyright © 2017 by Pointer Telocation, Ltd.

After launching the CAN Editor, it reads the current *.pl file from the Cellocator Programmer and displays the part relevant to the CAN parameters. You can edit this and save it in the *.pl file.

The CAN Editor also creates the **canconfig.xml** file using standard XML code. This file can be transferred to the server and is used for parsing the incoming Type 11 messages. When using the *Cellocator Integration Package*, the file will be stored by default on the server at **C:\ProgramData\Cellocator\Resources\XML\PL XML**. For more information about this file, refer to *canconfig.xml File Structure* on page 13.

The Default xml file released by Cellocator includes default FMS variables. These FMS variables are relevant only in J1939 mode.

3.3.2 XML Backend Management Practices

The structure of the Type 11 event, along with the XML file, allow the backend to understand clearly and uniquely, why the event was generated and what the CANBUS values in the message content are. For example, the explicit meaning of the trigger/s ID/s (The literal reason for the event generation) and explicit meaning of the various sensor IDs (the values in measurable units) sent in the message can be automatically deduced by the server side using the XML file which is generated by the CAN editor for each unique PL and saved on the backend. This XML is associated with a PL signature sent in every uplink message and can be used upon message reception for easy parsing and for the presentation layer update.

A TSP usually manages multiple vehicle fleets using a single server. In such a case, there will be a different configuration for each fleet or group of devices with a separate *canconfig.xml* file for each one. When a message is received, the server uses the PL signature provided in the message to identify the fleet or group of devices to which the unit belongs and then uses the relevant XML file to *decode* the message into an explicit application-level parameters, measurement units and values.

3.3.3 canconfig.xml File Structure – the Diagram.xml file

The Diagram.xml file (refer to Figure 5) is generated by the CAN editor and contains additional descriptive information not included in the binary configuration image that is downloaded to the remote units. This descriptive information is used by the server to parse the incoming Type 11 messages.

Copyright © 2017 by Pointer Telocation, Ltd.

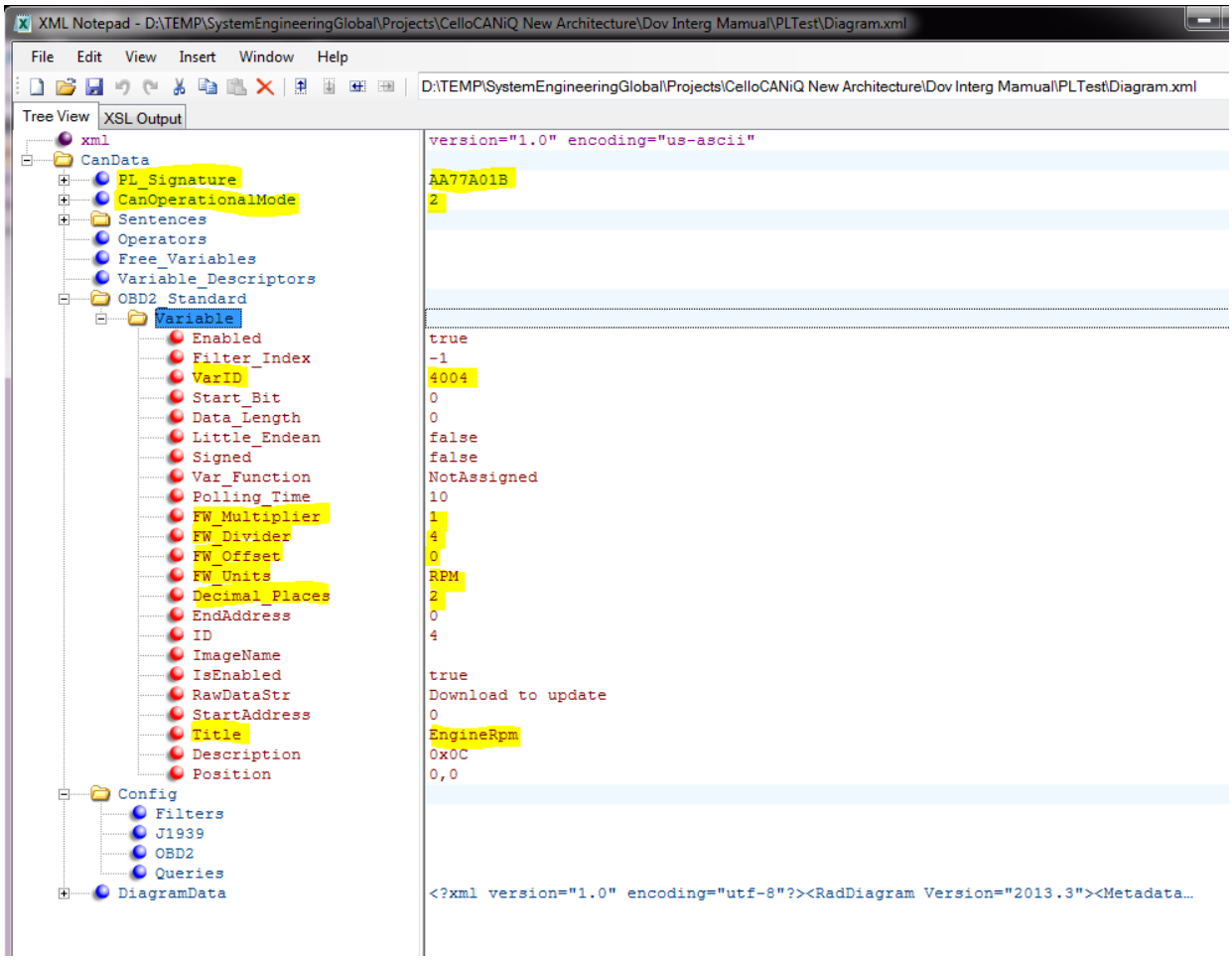
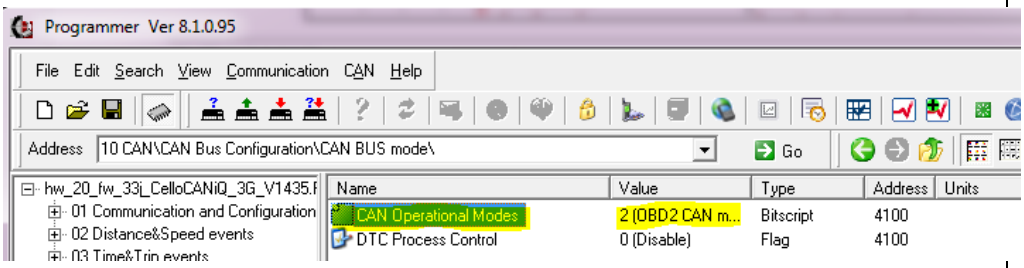


Figure 5 – Diagram.xml File

The following table describes the fields highlighted in this example and which are important for server-side handling:

Field Name	Description
PL Signature	The PL Signature is a unique 32-bit code identifying the PL. The PL signature is sent by the Cello-CANiQ unit via a Type 11 event message in module number 2 (Variable Dump List) and is used by the server application as a key to identify the matching XML file associated with the unit. Systems where the server manages multiple fleets can use the received PL signature to identify the originating fleet.

Copyright © 2017 by Pointer Telocation, Ltd.

<table>
<tr><td>CanOperational Mode</td><td>Represents the CAN operational mode as selected in the PL:
 

In this example, the selected operational mode is 2 for OBD2.</td></tr>
<tr><td>Variable</td><td>When a variable is selected by the CAN editor it becomes active (whether it is connected or not). All active variables are added to the PL and to the Diagram.xml file. The XML “variable” section includes all the variable properties needed for server-side parsing and presentation.</td></tr>
<tr><td>VarID</td><td>This is one of the “Variable” field properties in the XML file. The VarID is a unique number assigned to the variable during the CAN editing session.

A Type 11 message received with module 2 attached has the following structure:

<table>
<tr><td>0</td><td>Module Name 2- Trigger event</td></tr>
<tr><td>1</td><td rowspan="2">Length of module (16 bits)</td></tr>
<tr><td>2</td></tr>
<tr><td>3</td><td rowspan="2">Operator ID: 16 bits</td></tr>
<tr><td>4</td></tr>
<tr><td>5</td><td rowspan="4">PI Signature as described in Cellocator Wireless Communication Protocol.</td></tr>
<tr><td>6</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td></tr>
<tr><td>9</td><td>Number Of Variables</td></tr>
<tr><td></td><td>Variable ID (2 Bytes)</td></tr>
<tr><td></td><td>Variable Length</td></tr>
<tr><td></td><td>Variable payload</td></tr>
</table>
The server needs to parse the incoming message, access the associated XML file using the “PL Signature”, and use the message’s “Variable ID” field to find the “VarID” properties in the XML file. After this, the server can display the true variable value after appropriate value scaling. The following table shows a typical presentation layer handling for the “EnagineRPM” variable:

<table>
<tr><th>Field Name from XML</th><th>Description</th></tr>
<tr><td>VarID</td><td>4004 (Hex)</td></tr>
</table></td></tr>
</table>

Copyright © 2017 by Pointer Telocation, Ltd.

	Multiplier	These values define the linear scaling formula needed in order to transform the raw data sent by the unit to the presentation layer resolution. In this example, the EngineRPM needs to be divided by 4 to get the resolution of "Revolutions Per Minute" $RPM\ to\ be\ displyed = Offset + Var\ Payload \cdot \frac{Multiplier}{Divider}$ $= 0 + Var\ Payload \cdot \frac{1}{4}$ The RPM will be displayed with precision of 2 decimal digits as defined in the "Decimal Places" field in the XML.
	Divider	
	Offset	
	Decimal Places	
	Title	The "Title" field defines the name of the variable, in this case, "EngineRPM".

Copyright © 2017 by Pointer Telocation, Ltd.

4 Server-Side Integration

4.1 Integration Modes

Integrators can choose either of two possible integration modes:

- **Develop a new / improved Gateway** - The server must support the Type 11 protocol in addition to the existing fleet management protocol. This mode of integration requires full understanding of the Type 11 wireless protocol and requires server-side implementation of Type 11 protocol message parsing. Alternatively, integrators can modify an existing server.
- **Use the *Cellocator Integration Package*** – This implements full Type 11 protocol integration with the database. In this mode, integrators are required to implement database level integration between their application layer and the units represented as data-base entries.
  Refer to Cellocator Integration Package.

4.2 Generic Type 11 Handler Block Diagram

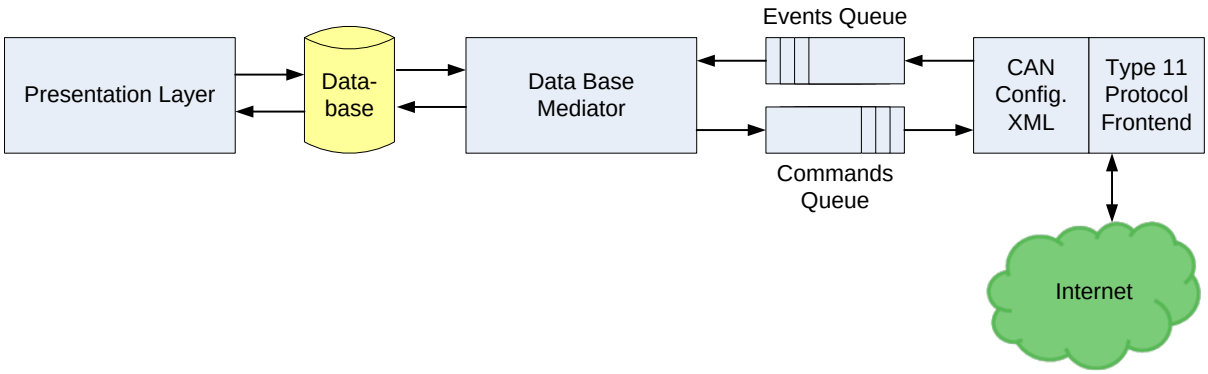


Figure 6 - Type 11 Server Building Blocks

The block diagram describes a generic Type 11 server application intended to manage multiple Cello-CANiQ units. The Type 11 protocol frontend service handles multiple sockets connected to Cello-CANiQ units.

When a Cello-CANiQ unit connects to the Type 11 server on the first occasion, an IPUP message is generated to register the unit's IP address and Unit ID in the server database. Type 11 datagrams received via open sockets are first checked to verify reception message integrity. If this is successful, a Type 11 acknowledgement message is sent to the unit to confirm message reception. Application messages are parsed, classified and then queued in the Received Events Queue. The Received Events Queue is read by the communication gateway and forwarded to the database which holds the history of all the Type 11 events.

In the downstream path, actions originated by the presentation layer (or automatic database routines) are written into a dedicated table in the database. The table update event is sensed by the communication gateway and an event is queued in the Type 11 protocol frontend. The Type 11 protocol frontend builds the protocol datagram and sends it to the socket associated with the destination unit ID.

Copyright © 2017 by Pointer Telocation, Ltd.

Refer to *OTA Interface Benefits* on page 19 for a general description of Type 11 benefits.

Refer to Cellocator Wireless Communication Protocol for a detailed description of the Type 11 protocol.

4.2.1 Typical Type 11 Scenarios

Figure 7 describes a typical upstream event. The Cell-CANiQ device registers a CAN trigger, sends a Type 11 message to the server, and the server acknowledges receipt of the message.

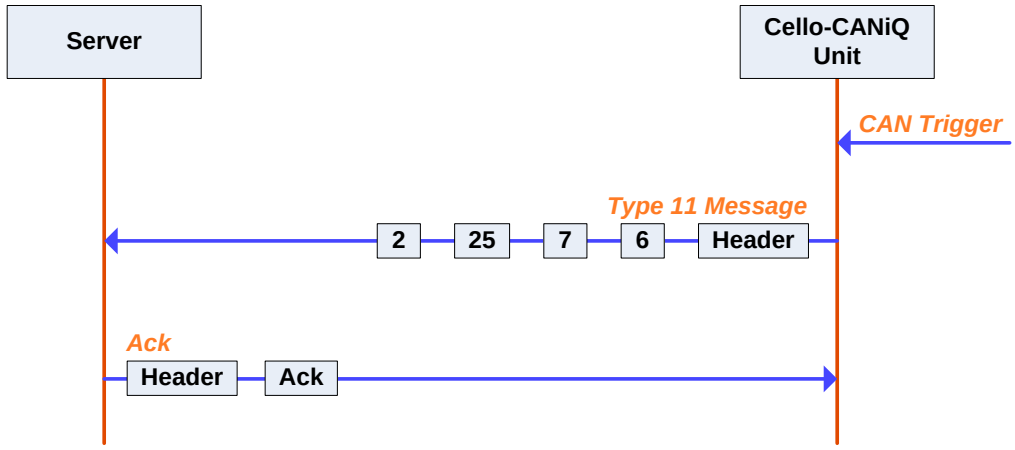


Figure 7 – Type 11 Scenario – Upstream Event

Figure 8 describes a typical downstream event. The server sends a "General Modules Query" Request asking for "CAN Variables Status dump". The Cello-CANiQ Unit will respond by sending back the values of all the CAN variables.. For more details please refer to the wireless Communication protocol section 3.8.5.16 "General Module Query" section.

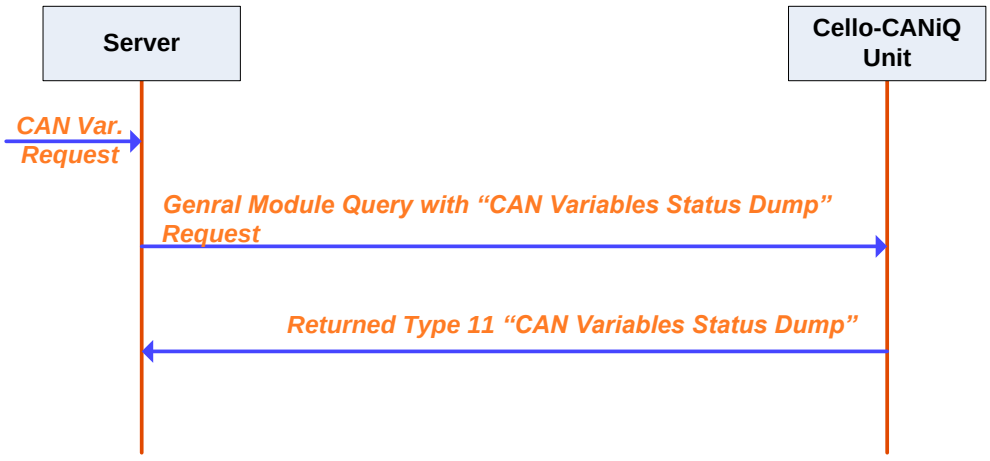


Figure 8 – Type 11 Scenario – Downstream Event

Copyright © 2017 by Pointer Telocation, Ltd.

4.3 OTA Interface Benefits

4.3.1 Features of the New Modular Type 11 Protocol

The Type 11 fleet protocol extends the functionality of Type 9 protocol by supporting larger data and configurable application message structures. Type 11 can also access multiple memory spaces, which previous protocols could not do.

Type 11 solves the problems of previous protocols, provides greater flexibility, and increases reporting capabilities. The following figures describes the structure of the Type 11 protocol.

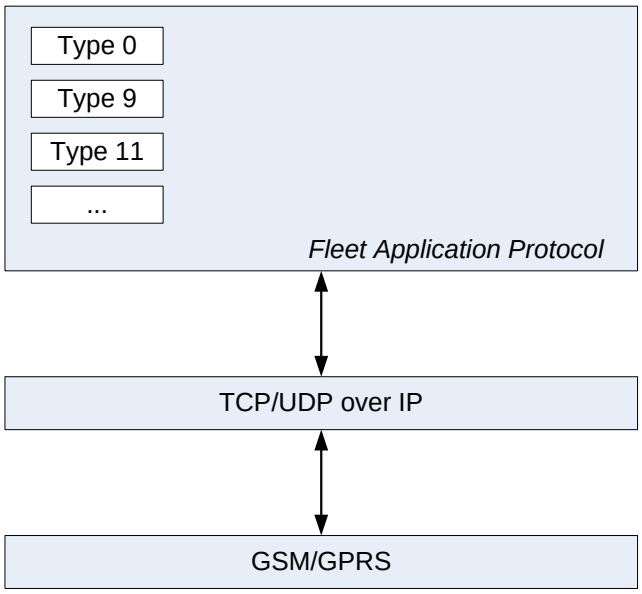


Figure 9 – Type 11 Protocol Structure

The following is a list of major Type 11 protocol features:

- The Type 11 data structure includes a new header format with 16 bits allocated for message length.

Copyright © 2017 by Pointer Telocation, Ltd.

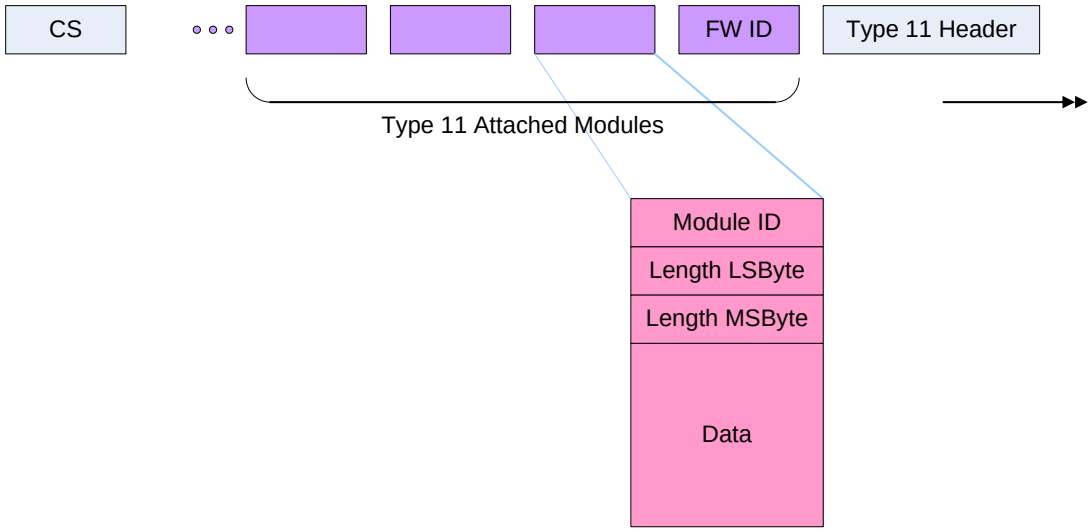


Figure 10 – Type 11 Protocol Data Structure

Each Type 11 module includes: Module ID, Length and Payload bytes.

- Type 11 enables users to program custom message structures. This information is stored in the *canconfig.xml file* which is then sent to the system server and used for parsing the incoming messages.
- Type 11 messages are logged messages. This means that after a message has been sent by a remote unit, it is saved in the non-volatile log memory until the server acknowledges receipt of the message. If necessary, the message is sent more than once until the server acknowledges its receipt, after which it is removed from the log memory. Logged messages are also known as guaranteed-delivery messages.

4.4 Type 11 Protocol Structure

This section provides basic information. For detailed information on the modules themselves, refer to the *Cellocator Wireless Communication Protocol* document.

4.4.1 Message Components

Type 11 is a true modular protocol. It is designed to carry records with a predefined structure, called modules. The protocol is used to extend the legacy fleet Cellocator OTA protocol.

Type 11 supports a theoretical message length of up to 65536 bytes, though in practice, this is limited by hardware limitations. It contains the following data (listed in the actual transmission order):

- System code – 4 bytes
- Message type – 1 byte
- Target Unit's ID – 4 bytes
- Communication Control Field - 2 bytes
- Packet Control Field - Legacy Fleet Field – 1 byte
- Message Length – 2 bytes
- Spare Bytes – 4 bytes
- Payload Modules – depends on user configuration – The available modules are listed in: Summary of Supported Type 11 Modules.

Copyright © 2017 by Pointer Telocation, Ltd.

- Error detection code – 8-bit additive checksum (excluding system code)

4.4.1.1 General Outbound Type 11 Message Format

The following table gives details of the Type 11 outbound message structure:

1	System Code, byte 1 – ASCII "M"
2	System Code, byte 2 – ASCII "C"
3	System Code, byte 3 – ASCII "G"
4	System Code, byte 4 – ASCII "P"
5	Message Type byte (11)
6	Unit's ID (total 32 bits)
7	
8	
9	
10	Communication Control Field (2 bytes)
11	
12	Command Numerator
13	Packet Control Field – Legacy fleet field
14	Length (of the modules section - not including the checksum)
15	
16	Spare (sent as 0)
17	
18	
19	
20	FW_HW ID Module : Mandatory
…	
	Other modules
	Check Sum

4.4.1.2 General Inbound Type 11 Message Format

The following table gives details of the Type 11 inbound message structure:

1	System Code, byte 1 – ASCII "M"
2	System Code, byte 2 – ASCII "C"
3	System Code, byte 3 – ASCII "G"
4	System Code, byte 4 – ASCII "P"
5	Message Type byte (11)
6	Destination Unit's ID (total 32 bits)
7	
8	
9	
10	Numerator
11	Authentication
12	
13	

Copyright © 2017 by Pointer Telocation, Ltd.

14	
15	Packet Control Field
16-17	Length (of the modules section - not including the checksum)
18 19 20 21	Spare (sent as 0)
20	Modules
	Check Sum

4.4.2 Type 11 Module Structure

The following table gives details of the Type 11 module structure:

0	Module Code
1 2	Length of module (16 bits) – Number of payload bytes
3 n	Payload bytes depended on the module code . . . .

Copyright © 2017 by Pointer Telocation, Ltd.

4.4.3 Summary of Supported Type 11 Modules

The following table lists the supported Type 11 modules:

Number	Description	Inbound	Outbound
0	Spare		
1	DTC Event Module		√
2	Trigger Event Module (Programmable Message)		√
3	Spare		
4	Calibration Data Snap Shot Module		√
5	Spare		
6	GPS Location Stamp Module		√
7	GPS Time Stamp Module		√
8	Firmware ID Module		√
9	Ack Module		√
10	Configuration Memory Write Module	√	√
11	Configuration Memory Block Request Module	√	√
12	Spare		
13	Spare		
15	Spare		
16	Spare		
17	Spare		
18	Spare		
19	Spare		
20	Spare		
21	VIN String Write Module	√	
22	VIN String Request Module	√	√
23	Infrastructure – Calibration command Module		
24	Spare		
25	Trigger Event ID Module		√
26	(Infrastructure)		
27	Spare		
28	General Status Event		√
29	General Module Query (Inbound Only)	√	
30	PHSN_General_Status_Event	√	√
31	CAN Variables Status Dump (OutBound)		√
32	General Command (InBound)	√	

Copyright © 2017 by Pointer Telocation, Ltd.

4.5 Automatic GNSS Antenna Status

GNSS antenna status indications are sent via a Type 0 message in the Service and Location Status Byte (byte 41, bit 0).

- Selected antenna is reported in the fleet type 0 message

Functions as the upper bit selecting the role of bytes 33 to 38 (CR200)	IMEI Bit 49	IMEI Bit 48	CFE Type			Trailer status indication: 0-Trailer disconnected 1-Trailer connected	Actual GNSS antenna selected 1-internal 1-external
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

The GNSS antenna mechanism supports the following:

- External antenna fault detection (for example: shorted or disconnected)
- Automatic internal or external antenna selection for optimal GNSS reception

Copyright © 2017 by Pointer Telocation, Ltd.

5 Cellocator Integration Package

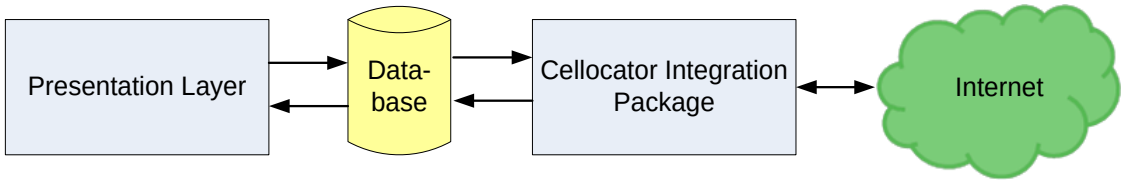


Figure 11 - Cellocator Integration Package

The Cellocator Integration Package is a software solution created by Pointer for server-side developers wishing to achieve database integration level between their application layer and Cello family units. The Cellocator Integration Package defines the database table structures and commands needed to manage multiple Cello family units in a balanced, modular and expandable environment. The package was updated to support the Type 11 protocol on top of the legacy fleet management. For more detailed information about database level integration using the Cellocator Integration Package, refer to the Cellocator Integration Package [Full Edition] Manual.

Copyright © 2017 by Pointer Telocation, Ltd.