Cellocator Integration Tool Guide

Cellocator Division
Pointer Telocation Ltd.

Proprietary and Confidential

Version 1.0a

Revised and Updated: April 6, 2021

POINTER

Pointer Telocation Ltd. 14 Hamelacha St., Rosh Ha'ayin 48091, Israel • Tel: 972-3-5723111 • Fax: 972-3-5723100 • www.pointer.com

Legal Notices

IMPORTANT

1. All legal terms and safety and operating instructions should be read thoroughly before the product accompanying this document is installed and operated.
2. This document should be retained for future reference.
3. 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.

Intellectual Property

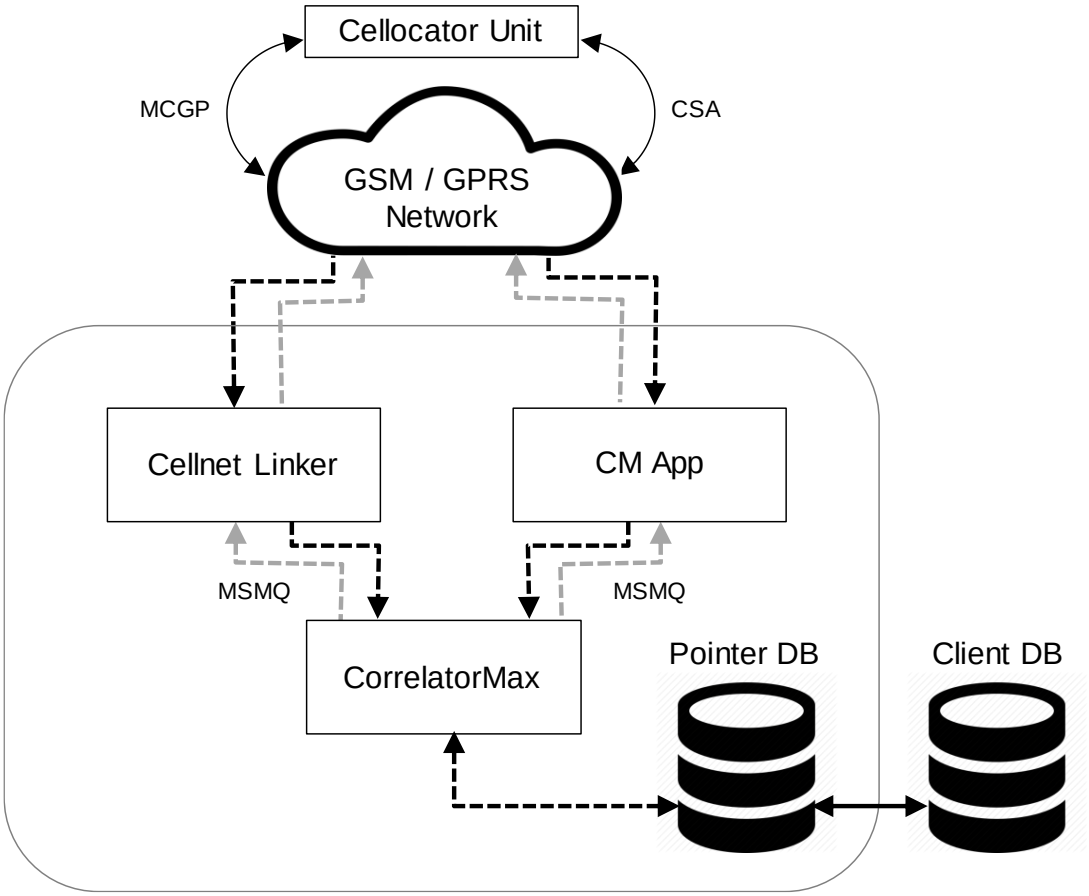
Copyright in and to this document is owned solely by Pointer Telocation Ltd. Nothing in this document shall be construed as granting you any license to any intellectual property rights subsisting in or related to the subject matter of this document including, without limitation, patents, patent applications, 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.

Table of Contents

1 Overview

The purpose of the Cellocator Integration Tool is to provide customers wishing to integrate the Cellocator Over-The-Air (OTA) protocol into their production environment with a complete and quick to implement solution.

This document provides a complete product description of the Integration Tool solution and other integration related information, for the purposes of integrating the Cellocator OTA protocol within a client's production environment.



1.1 Enhancements in this version

This section describes the main enhancements added in version 2.7.297 of the Cellocator Integration Tool. These enhancements include:

- **Process Monitor improvements:** The Pointer Monitor Service is now a standard Windows service, configured to start upon system server/OS start and administrated via the Windows management console and other Windows administration tools, as per any regular Windows service. The Pointer Monitor Service is also now part of the default installation package, as described in *Performing the Installation*.
- **Buffering mechanisms are now unified:** The Cellnet Linker and CM App now both use MSMQ as their queueing mechanism.
- **Installation and upgrade process improvements:** The installation process has been enhanced to include the ability to delete an existing database when reinstalling the Cellocator Integration Tool. The installation process for database tables has also been improved to ensure a smoother installation.
- **CorrelatorMax message processing improvements:** CorrelatorMax output (data inserted into the store and forwarded towards the application) has been improved in its presentation and for ease of use, including improved support for server backward compatibility.
- **Cellnet Linker license mechanism removed:** A licensing mechanism which stopped the Cellnet Linker from working after 3 years has been removed.
- **MySQL compatibility:** The Cellocator Integration Tool now also supports MySQL, in addition to Microsoft SQL.
- **Logical support for downlink messages:** A new API has been implemented to free customer applications from handling the row data format of downlink messages. To maintain backward compatibility, the new mechanism allows previously integrated customer services to use the old mechanism.
- **CANBUS messages logical parsing:** A new consistent mechanism of CANBUS messages parsing has been implemented, together with a unified protocol interface. This ensures Cellocator devices can correctly interpret CANBUS messages which are the result of non-standard CANBUS protocols and environments, namely different vehicle models, manufacturers, and engine type. There are two modes of implementation: a **new mode** and **past mode** (for backward compatibility).
- **Documentation improvements:** To assist customers in their integration efforts, the Cellocator Integration Tool Guide (this document) has been completely revised.

1.2 What is in The Box?

The Cellocator Integration Tool is comprised of four applications (**Cellnet Linker**, **CorrelatorMax**, **CM App** and **Pointer SMS Gateway**), the **Pointer Monitor Service**, and **Microsoft SQL / MySQL database objects**.

This section describes the role of each application, its installation options, and the way it communicates with its next tier application. For a visual representation of the role that each of the components plays in the integration process, see *Architecture* .

1.2.1 *Cellnet Linker*

Cellocator's Cellnet Linker application is a 1st tier GPRS communication application written in the C# .Net platform and supports the MCGP OTA protocol. By default, it is installed as part of the install setup wizard (see *Performing the Installation*).

The Cellnet Linker communicates directly by sending downlink commands and processing uplink messages from Cellocator units, while communicating with the CorrelatorMax application using MSMQ.

The Cellnet Linker enables bi-directional communication with Cellocator units, including IP/Port/Socket management, monitoring, and other management features.

1.2.2 *CorrelatorMax*

Cellocator's CorrelatorMax application is a 2nd tier GPRS SMS communication application written in the C# .Net platform. By default, it is installed as part of the install setup wizard (see *Performing the Installation*).

The CorrelatorMax communicates with the Cellnet Linker application using Microsoft Messaging service (MSMQ) and with the CM App for CSA protocol via the MSMQ mechanism.

This layer of communication ensures the sending of downlink messages and the receiving of uplink messages to/from Cellocator units, while communicating with Microsoft SQL Server / MySQL where it stores parsed uplink messages and retrieves downlink messages designated for Cellocator units.

1.2.3 *CM App*

Cellocator's CM App is a 1st tier GPRS communication application written in C# .Net platform and supports the CSA OTA protocol. By default, it is installed as part of the install setup wizard (see *Performing the Installation*).

The CM App communicates directly by sending downlinks commands and processing uplinks messages which are CSA related from Cellocator units, while communicating with the CorrelatorMax application using the MSMQ service.

Note that both the Cellnet Linker and CM App can work simultaneously with the same devices on different ports on the same server.

1.2.4 *Pointer SMS Gateway*

The Pointer SMS Gateway (PSG) application is a 1st tier SMS communication application written in the ASP.Net platform. By default, it is installed as part of the install setup wizard (see *Performing the Installation*).

The PSG communicates directly (the PSG uses a 3rd party application to send/receive SMS, as described in *Configuring Support for Bi-Directional SMS Communication*) by sending downlink commands and processing uplink messages from Cellocator units while communicating with the CorrelatorMax application using the Microsoft Messaging service (uplinks) and HTTP requests (downlinks).

1.2.5 *Pointer Monitor Service*

The Pointer Monitor Service is a standard Windows service configured to start upon every system server/OS start. The Pointer Monitor Service is included in the install setup wizard (see *Performing the Installation*), and you can also determine if the Service should launch upon completing the installation.

As it is a standard Windows Service, the Pointer Monitor Service complies with regular rules and protocols and is run via the Windows Management Console and other Windows administration tools. It also manages the other components of the Cellocator Integration Tool (Cellnet Linker, CM App and CorrelatorMax), meaning that it restarts the applications upon crashes and failures, and launches the applications upon system start or restart.

1.2.6 *Database Objects*

Cellocator utilizes Microsoft SQL and My SQL Server technology to communicate with the CorrelatorMax application. By default, Database Objects are installed as part of the install setup wizard (see *Performing the Installation*).

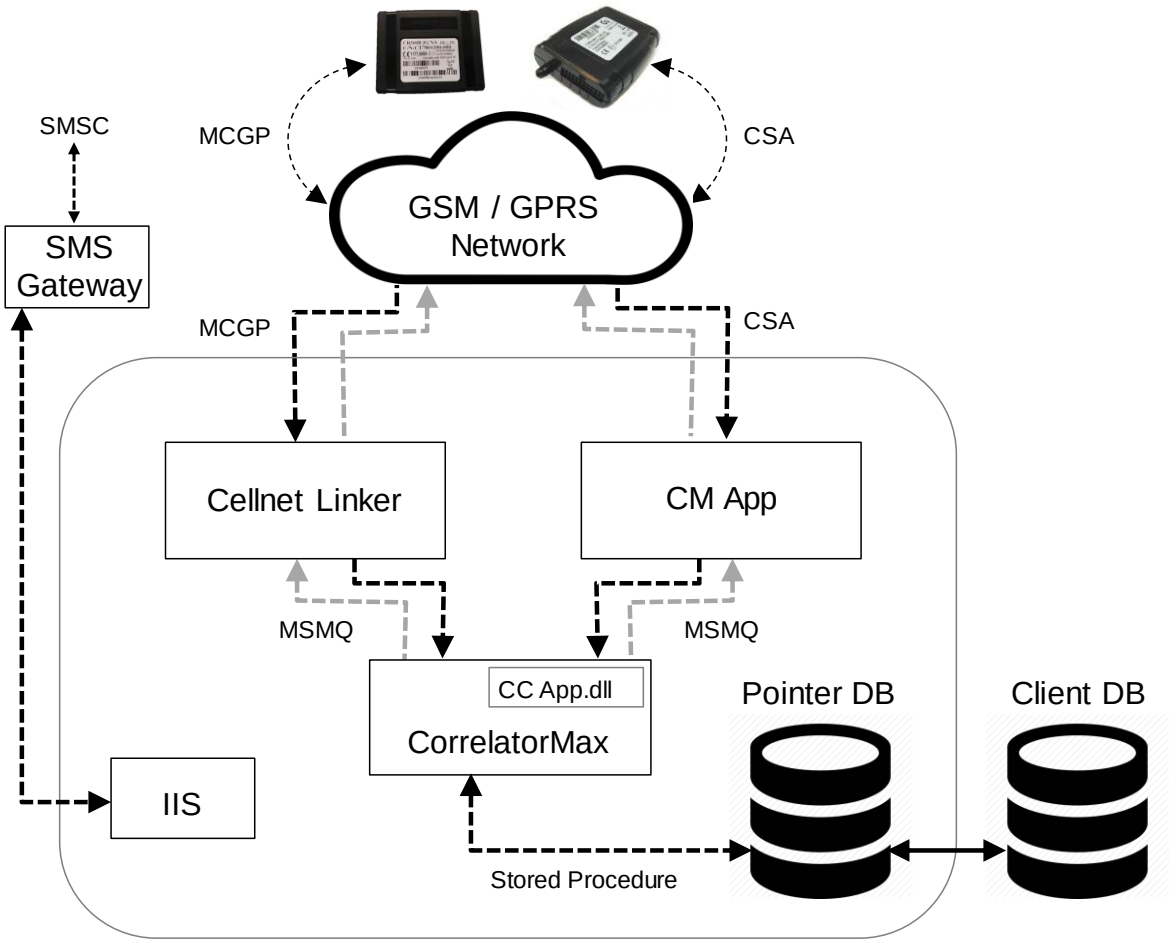
Communication is done via a set of database tables and stored procedures. There are three tables in the database:

- In the downlink flow: The **CMUDLQueue** table, which hosts both protocols (MCGP/S and CSA).
- In the uplink flow: The **UPlinkMsgLog** table contains all MCGP/S messages, and CSA (Safety) messages which include GPS values (such as full events), and the **ModularLog** table, which contains all CSA messages (including a reference to UplinkMsgLog records), as well as CAN and Nano messages.

You can also build your own set of tables for Cellocator uplinks and downlinks and utilize the store procedures accordingly to pull and place uplinks and downlinks.

1.3 Architecture

The following illustration shows the role of the Integration Tool's main components in both the uplink flow (from the Cellocator unit to the database) and downlink flow (from the database to the unit). For further information, see *Uplink / Downlink Message Flow*.



1.3.1 *Uplink / Downlink Message Flow*

1.3.1.1 The Uplink Message Flow

The uplink message flow originates from the Cellocator unit (for both Fleet applications and CSA supported units; this section describes both flows in the following steps) and terminates in the database.

1. The Cellocator unit sends uplink messages using the TCP or UDP protocol to the Cellnet Linker (for Fleet applications) or CM App (for Safety applications).
2. The Cellnet Linker / CM App receives the uplink message and sends an acknowledge message back to the Cellocator unit.
3. *(For Fleet applications)* The message is then wrapped with the Pointer MSMQ format and sent to the CorrelatorMax input queue.

(*For CSA applications*) The message is then transferred to the CorrelatorMax via bidirectional MSMQ using the **CC App dll**.

4. *(For Fleet applications)* The CorrelatorMax pulls the uplink message from its input queue and extracts and parses it according to the Cellocator MCGP OTA protocol.

   (*For CSA applications*) The CorrelatorMax (CC App dll) pulls the uplink message from its MSMQ and extracts and parses it according to the Cellocator CSA OTA protocol.

5. The CorrelatorMax inserts the uplink message to the **UpLinkMsgLog** table in the database server using the **spCMUProcessUnitUplinkMsg** store procedure.

1.3.1.2 The Downlink Message Flow

The downlink message flow originates from the database server (for both Fleet applications and CSA supported units; this section describes both flows in the following steps, in addition to SMS messages) and terminates at the designated Cellocator unit.

NOTE: The CSA protocol does not support SMS.

---

1. *(For Fleet applications)* Insert a MCGP command to the Cellocator unit through the **CMUDLQueue** table in the database server using the **spCMUDLQueue_Get** stored procedure.

   (For CSA applications) Insert a CSA command to the Cellocator unit through the **CMUDLQueue** table in the database server using the **spCMUDLQueue_Get** stored procedure.

2. The CorrelatorMax pulls the downlink message from the table and builds the command according to the Cellocator MCGP/S OTA protocol or CSA OTA protocol.

3. *(For Fleet applications)* The CorrelatorMax wraps it in Pointer MSMQ format and sends it to the Cellnet Linker input queue. The Cellnet Linker then pulls the message from its input queue, unwraps it and sends it to the specific Cellocator unit.

   (For CSA applications) The CorrelatorMax sends it to the CM App via a dedicated MSMQ, using the **CC App dll**. The CM App then pulls the message from the MSMQ and sends it to the specific Cellocator unit.

   (For SMS commands) The CorrelatorMax wraps it in an HTTP request to send to the Pointer SMS Gateway (PSG). The PSG processes the message and sends it to the unit (using the installed third-party SMS broker).

NOTE: SMS downlinks can be sent in two ways:

- By inserting an SMS command through the **CMUDLQueue** (the NetworkType field should be set to 5).
- Automatically via the CorrelatorMax (if the **SMS_Replay** value in the Correlator INI is enabled). If a GPRS command is sent to a unit and the unit does not reply to the command, the Cellnet Linker issues a report to the CorrelatorMax (Error 10002) and the CorrelatorMax sends the same command by SMS upon receiving such a report. The unit will reply to this command by SMS too, so bi-directional SMS support is required.

---

4. The Cellocator unit receives the downlink message, verifies the command, and sends an acknowledge/request message back to one of the CM App (for CSA applications), the Cellnet Linker (for Fleet applications), or the PSG (for SMS commands).

1.4 Compatibility / Scalability

For full details of the compatibility of each of the Integration Tool components with your environment, see the relevant section under *Installing and Configuring the Individual Components*.

1.4.1 *Integration Tool OTA Compatibility*

The Integration Tool 2.7.279 is compatible with the Cellocator OTA protocol (both MSGP/S and CSA), as specified in the table below. Note that CSD communication is not supported.

MCGP & MCGS OTA Protocol			
No.	**Message Direction**	**Protocol Type**	**Description**
1	Uplink message	Message type 0	Status message marked as Logged message or Emergency message.
3	Downlink message	Message type 0	Generic command message.
4	Downlink message	Message type 1	Programming command message.
5	Uplink message	Message type 3	Programming data message (usually Ack. for programming command).
6	Downlink message	Message type 5	Forward data message (MDT\Garmin\Transparent).
	Uplink message	Message type 7	Logged fragment of forwarded data (serial devices).
7	Uplink message	Message type 8	Forward data message (MDT\Garmin\Transparent).
8	Downlink message	Message type 9	Modular command message.
9	Uplink message	Message type 9	Receiving Modular Sub Type X CANBUS messages when set to be sent with GPS data and time stamp. Note only the GPS and time stamp data are parsed. The CAN sensors are not parsed.
10	Downlink message	Message type 11	Can IQ Modular command message, Nano & MultiSense.
11	Uplink message	Message type 11	Receiving CAN IQ Modular Sub Type X messages.
CSA OTA protocol			

No.	Message Direction	Protocol Type	Description
1	Uplink message	Message type 0	CSA event / reply to command.
2	Uplink message	Message type 3	Programming data message (usually Ack. for programming command).
3	Downlink message	Message type 1	Ack (Acknowledgment to message type 0).
4	Downlink message	Message type 2	Programming commands.
5	Downlink message	Message type 4	CSA commands – requests.

1.4.2 *Integration Tool Scalability*

The Integration Tool can support advanced deployment configurations that need to handle high load environments as well to provide a fail-over solution. The scalability options are listed below.

- The Cellnet Linker application can be installed in several instances on the same server to support operations in TCP or UDP ports simultaneously, working with different port numbers for different types of applications, and more.
- The Cellnet Linker can also be installed on several servers with a Network Load Balancing service or appliance to utilize load balancing and high availability environments.
- The CM App can be installed in several instances on the same server to support operations in TCP or UDP ports simultaneously, working with different port numbers for different types of applications, and more.
- The CorrelatorMax can work with several Cellnet Linker applications installed on a single server or spread across several servers, using the MSMQ service.

1.5 Terminology

The following table lists the common terms and acronyms found in this document.

Term	Description
Uplink message	Message which originates from the Cellocator unit and is sent to the GPRS/SMS gateway.
Downlink message	Message which originates at the GPRS/SMS gateway and is sent to the Cellocator unit.
MSMQ	Microsoft Message Queuing
GPRS	General Packet Radio Service
CCC	Command & Control Center
MCGP	Main Cellocator Fleet management OTA protocol
CSA	Cellocator Safety Application OTA protocol

Term	Description
SMSC	Short Message Service Center; a network element in the mobile telephone network which stores, forwards, converts and delivers Short Message Service (SMS) messages.

2 Installation

This chapter describes how to install the Cellocator Integration Tool and all its components, and includes the following sections:

2.1 Prerequisites

- Windows 10/Windows 2016 Server or higher with the latest Service Pack.
- Microsoft.NET Framework 4.6.
- MSMQ service.
- IIS 5 or above (if the SMS Gateway is installed).
- SMS Broker Application (if the SMS Gateway is installed).
- Microsoft SQL Server 2016 or higher (English only).
  - SQL Server Authentication: SQL Server.
  - SQL user with SQL Authentication and Sysadmin permission.
- ODBC Driver version 17 (for SQL server 2019 (15.0.2000.5)) must be installed on the machine before Integration package installation.

  If the SQL Server is on a remote machine, SQL Client tools MUST be installed on the Integration Tools server.

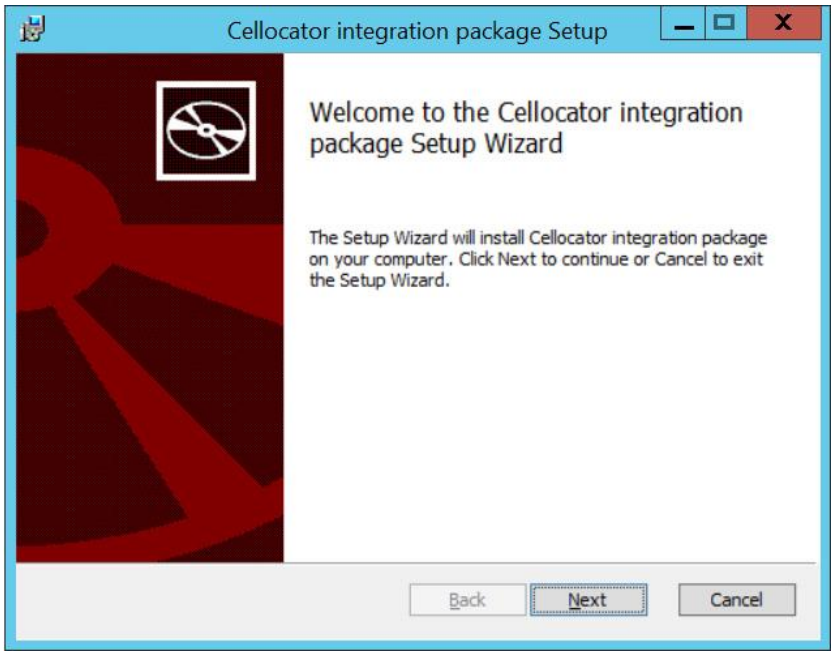
IMPORTANT: The installation of this Integration Tool package configures the INI files of each component assuming all are running on the same server. If you plan to install it on separate computers, you will need to modify the configuration files accordingly. For further information about the relevant settings that need to be changed, see the *Installing and Configuring the Individual Components* section.

2.2 Performing the Installation

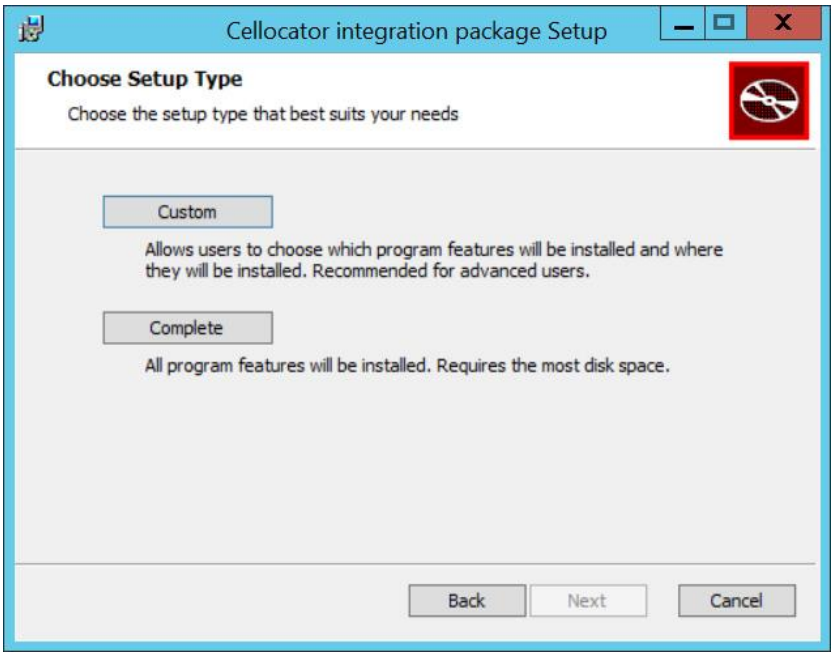
The following procedure describes how to install the Cellocator Integration Tool, which includes all the individual components by default; for details on how to install and configure each of the components separately, see *Installing and Configuring the Individual Components*.

➤ **To install the Cellocator Integration Tool**

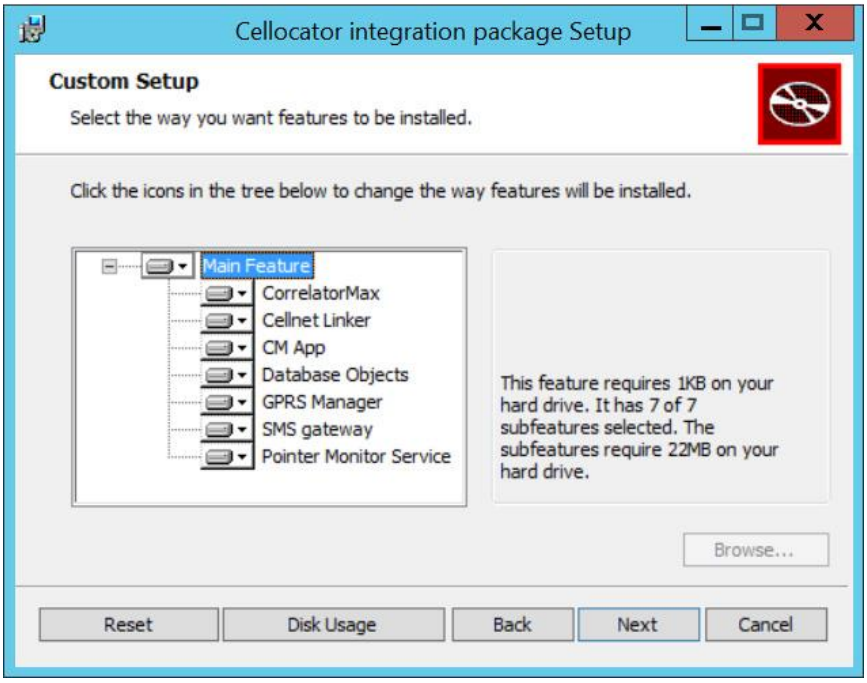
1. Double-click the **Integration Tool installer.msi** and in the displayed Setup Wizard screen, click **Next**.



2. In the displayed Choose Setup Type screen, select **Custom** or **Complete** and click **Next**.



Note that you can select **Custom** to install each component separately (click **Main Feature** to display the list of additional components, as shown below).



IMPORTANT: Selecting **Complete** will install all the components, including the CM App and SMS gateway. Select **Custom** if you don't want to install the CM App (For safety data only) or the SMS gateway.

---

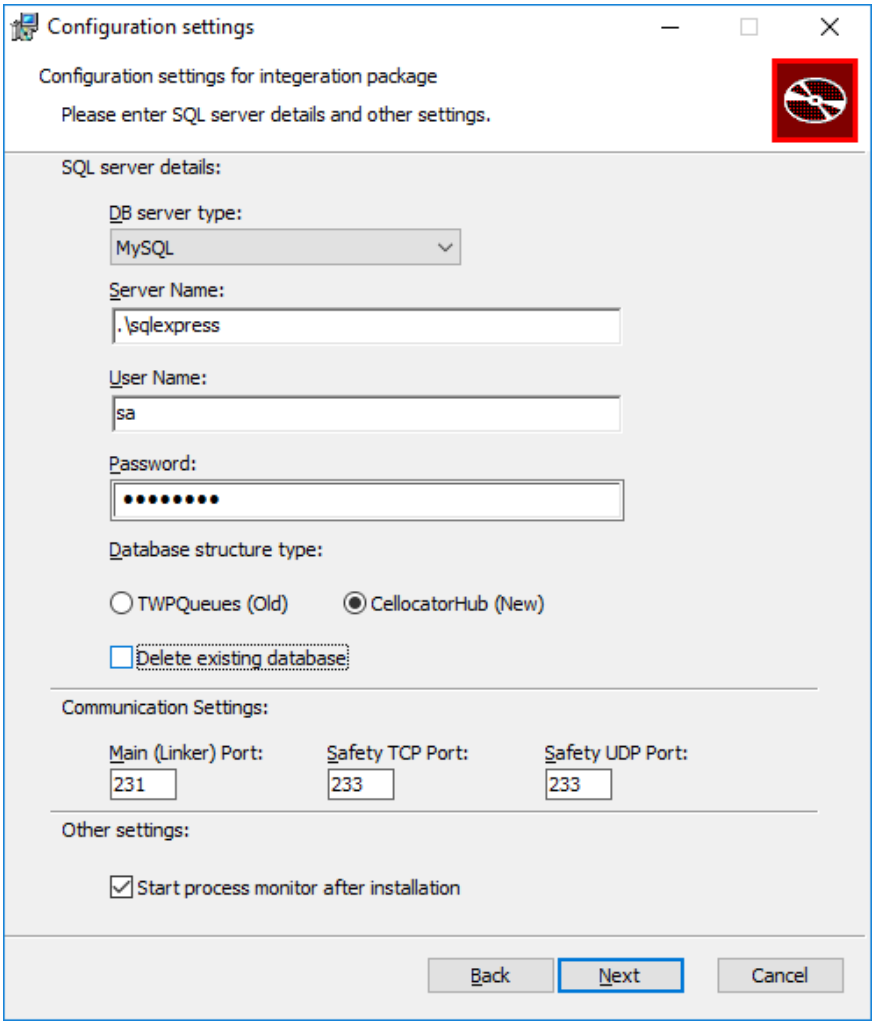
3. In the displayed Configuration Settings screen, define the following:
   - Select the **DB server type**; select either **Microsoft SQL** or **MySQL** from the option box.
   - Enter the SQL server credentials (**Server Name**, **Username** and **Password**).
   - Select the **Database structure type**; select either **TWPQueues (legacy)** or **CellocatorHub (New)**.

     Note that the **TWPQueues (legacy)** option enables existing customers to maintain the same database structure they have previously used in other integrations to ensure backward compatibility, while the **CellocatorHub** option will run the new and enhanced structure.
   - Select the **Delete existing database** checkbox if you want to remove an existing database. We recommend this option if you have a previous installation that you want to overwrite.

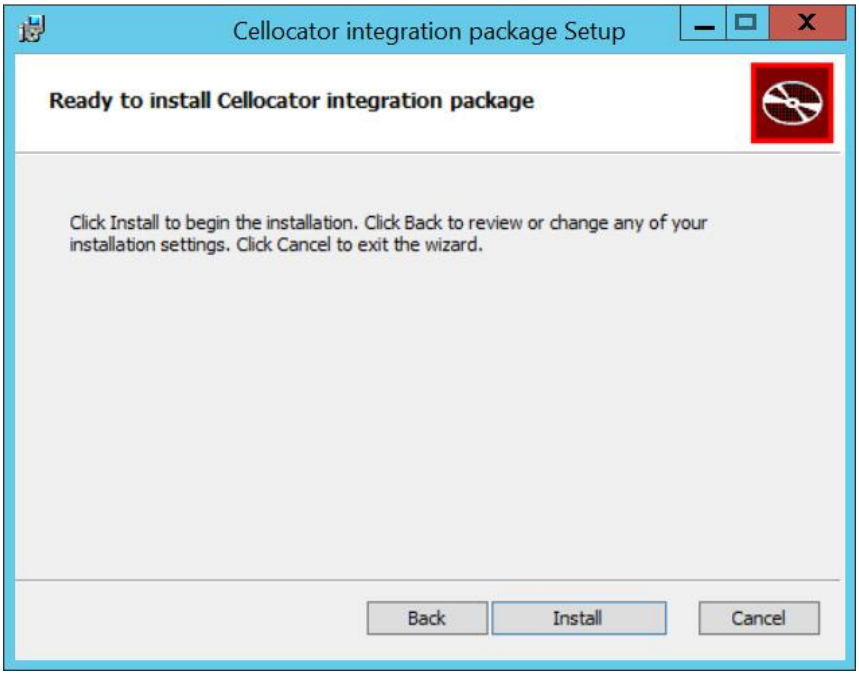
     Note that if you selected **Microsoft SQL** as the DB server type (see above), selecting to delete the existing database will back up the database to the default SQL Server backup folder; if you selected **MySQL**, the database will not be backed up.

- Define the relevant port settings in the **Communication Settings** section. These ports are for the Cellnet Linker (by default, port 231 is selected) and CM App applications (by default, port 233 is selected for both TCP and UDP).
- Select the **Start process monitor after installation** checkbox if you want to launch the Pointer Monitor Service upon completing the installation wizard. Note that you can always launch the Pointer Monitor Service manually from the command line, via the tray icon, or through the Windows Task Manager.

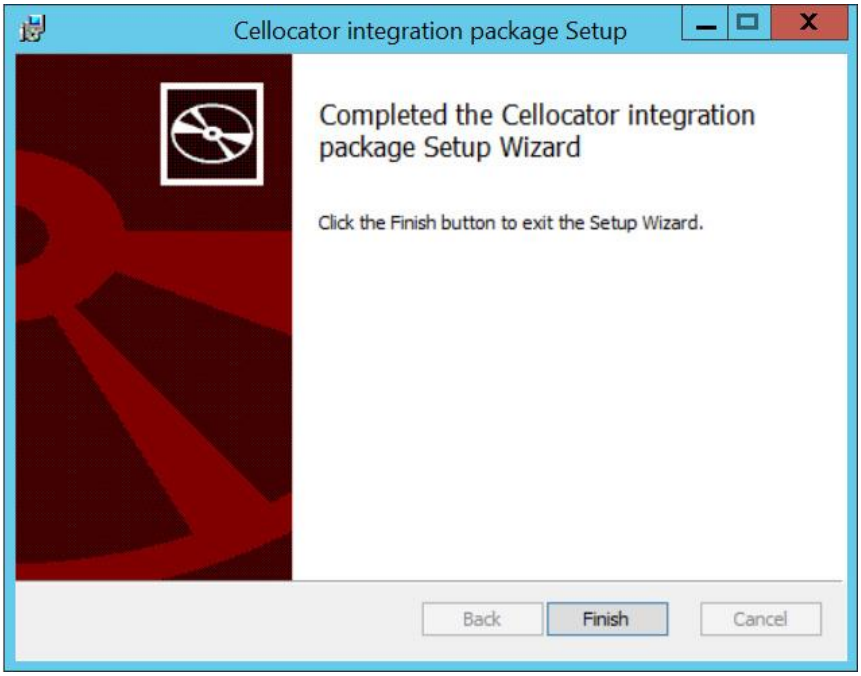
After defining the above settings, click **Next**.



4. In the displayed Ready to Install screen, click **Install** to install the package.



5. Once the installation is complete, click **Finish** to exit the Setup Wizard.



If you selected to start the Pointer Monitor Service in Step 3, the CorrelatorMax, Cellnet Linker, and CM App components are launched automatically.

If you did not select to start the Pointer Monitor Service after installation, you can start the service from the **Services** tab in the Windows Task Manager dialog box. Likewise, if you need to start or stop any of the components, use the Task Manager to stop and start the relevant service for each component.

- ➔ To verify the installation, see Verifying the Installation.
- ➔ If you plan to install the Integration Tool components on different computers, see Installing and Configuring the Individual Components.
- ➔ If you encountered issues with your installation, see **Error! Reference source not found.**.

2.3 Verifying the Installation

The following steps describe how to verify the installation of the Integration Tool package and to ensure you are ready to begin integrating.

1. Check that the Integration Tool package is in **C:\Program Files\Cellocator\Integration Package**.
2. Double-click the **CellNetLinker.exe**. Make sure the Cellnet Linker component launches with no errors.
3. Assuming CM App was installed, double-click the **CM App.exe** and make sure ports are available and that the application launches with no errors.
4. Double-click the **Correlator.exe** and make sure it launches without errors.
5. Assuming the SMS Gateway was installed, type this URL in the browser: **http://localhost/PSG/PSG.aspx**. The browser should show this message:

   ```
   "False The following parameters are missing: Message PhoneNum
   ApplicationType MsgDirection"
   ```

   Note that SMS Broker settings must be defined in the INI file. For further details, see *Installing and Configuring SMS Gateway*.
6. Connect to the SQL server and verify the creation of the database according to the TWPQueues or CellocatorHub structure you selected in the installation process.
7. Verify all DB tables were created.
   - CMUDLQueue
   - UplinkMsgLog
   - SMSBillingLog
   - SMSTransport
   - ModularLog
   - Commands
   - DownlinkQueue
   - UplinkMsgLog (column names were changed, and the modular messages are combined into this table)
8. Verify all DB stored procedures were created.
   - spCMUDLQueue_Get
   - spCMUProcessUnitUplinkMsg
   - spCMUs_Get
   - spInsertSMSBillingLog
   - spPAI_IsRawDataRequest
   - spCMUDLQueue_Get
   - spCMUProcessUnitUplinkMsg
   - spGenerateCommand
9. Verify the following functions were created.
   - fnSplitToString

- ➔ If the OSQL is not accessible, please check that you have data base engine installed!
  - OSQL version shall match to ODBC driver version.
- ➔ If the MSSQL server does not support Windows authentication.
  - Need to create predefined SQL authentication user and use it during installation.
- ➔ If your installation was successful, and you plan to install the Integration Tool components on different computers, see Installing and Configuring the Individual Components.
- ➔ If you need to configure SMS Broker applications to support SMS communications, see Configuring Support for Bi-Directional SMS Communication.

2.4 Installing and Configuring the Individual Components

This section describes how to install and configure each of the Integration Tool components separately for customized installation scenarios (such as the installation of components on different computers), and includes:

- **Installing and configuring Cellnet Linker**, see below.
- **Installing and configuring CM App**, page 27
- **Installing and configuring SMS Gateway**, page 30
- **Installing and configuring CorrelatorMax**, page 34
- **Installing and configuring Pointer Monitor Service**, page **Error! Bookmark not defined.**
- **Installing and configuring Database Objects / Server**, page 43

2.4.1 *Installing and Configuring Cellnet Linker*

2.4.1.1 Prerequisites

- Refer to the prerequisites in section 2.1.

2.4.1.2 Installing the Cellnet Linker

To install the Cellnet Linker, double-click the **CellNetLinker.exe** file and follow the onscreen instructions.

To verify the installation, ensure the following:

- Cellnet Linker installation finished successfully without errors.
- Cellnet Linker INI file is configured properly (see the following section for further details about configuring the INI file).
- Log folders were created successfully according to the INI file.
- Cellnet Linker can be launched successfully.

IMPORTANT: Cellnet Linker is a user space application and therefore a user session cannot be logged off, otherwise the application will be closed and no uplinks or downlinks will be processed by the application. We recommend you lock the user session to keep the application up.

2.4.1.3 Configuring the Cellnet Linker

Open CellNetLinker.exe.INI.asp file to edit and configure operational parameters according to the explanations provided in the following tables. Note that the parameters marked in blue must be changed; all other parameters can remain with the default values or can be changed according to your needs.

2.4.1.4 Application

Key	Valid value	Default	Description
Instance.Id	Number		Application instance ID. Must be unique (starting from 1) to differentiate between different instances of the applications.
Auto start delay (sec)			
Monitor auto.track			
Monitor max lines			
Monitor All	True/false	True	Indicates whether to show all events in the Linker window. When set to `false`, events will not be logged to the screen; when set to `true` all events will be logged in the screen.

2.4.1.5 Server

Key	Valid value	Default	Description
Port			
Message forwarding Enabled			Enable or disable uplink message forwarding. Works for TCP & UDP protocol.
Message forwarding IP			IP address of another Cellnet Linker instance for message forwarding.
Message forwarding Port			Port number of another Cellnet Linker instance which will listen to the forwarding messages.
Message forwarding Type	Number	-1	Controls and filters the message type the Linker will forward, e.g., 0 for type 0 or 8 for type 8 messages.
Allow Serve for All Clients	String	True	Allows all clients to connect to the Linker.
Serve Enabled	String	False	Enable/disable connection filtering.
Serve IP	String		Defines the allowed IP connections when serve is enabled.

2.4.1.6 Log file

Key	Valid value	Default	Description
`Path`		Application folder	Application log files folder, by default ./log.
`Max lines`	100 …. 109	10000	Max lines in log file.
`Max size (KByte)`	1024 ... 106	10240 [=10MB]	Max log file size, in KBytes [1K=1024].
`Max duration (Min)`	5 ... 525600	60 [= 1Hr]	Log file max duration in minutes.
`Buffer byte seperator`	True / False	True	Write hex buffers using blank as byte separator.
`Filter All`	True / False	False	Indicates whether to log all events in the Linker log file. When set to `False`, events will not be written to the log file; when set to `True` all events will be written to the log file.

2.4.1.7 MSMQ (In & Out)

Key	Valid value	Default	Description
`Verify queue`			Verifies that input MSMQ queue is defined and accessible to the application.
`Queue` (under `MSMQ.In` section)			Application input MSMQ queue using FormatName: 'DIRECT=TCP:<IP>\[Private$\]<Queue_name>', path should be the same as defined in the Correlator INI file, in the `MSMQ Out` section of the Cellocator paragraph. **NOTE**: The name of the Linker IN Queue should match the name of the Correlator OUT Queue. The Queue must be created manually in MSMQ.
`Queue.0`			Application Output MSMQ queue using FormatName: 'DIRECT=TCP:<IP>\[Private$\]<Queue_name>', path should be the same as defined in the Correlator INI file, in the `MSMQ In` section of the Cellocator paragraph. **NOTE**: The name of the Linker OUT Queue should match the name of the Correlator IN Queue. The Queue must be created manually in MSMQ.
`. . .`			
`Queue.9`			

2.4.1.8 MSMQ Monitor

Key	Valid value	Default	Description
Enabled	True / False	True	Enable / disable MSQM size monitoring.
Cycle Minutes	> 0 …	0.5	MSMQ size check cycle in minutes.
Enable Size	True / False	true	Monitor according to size.
Upper Threshold KB	Numeric	50	Upper threshold; if MSMQ's total size exceeds this value, incoming port is closed.
Lower Threshold KB	Numeric	20	Lower threshold; if MSMQ's total size exceeded, reopens incoming port upon reaching lower threshold.
Enable Count	True / False	true	Enable MSMQ message count monitoring using performance counter
Monitor Total Count	True / False	False	If set to `false`, each queue is monitored separately; if set to `true` the total message count of all defined queues is checked to ensure they are within threshold limits.
Count Lower Threshold	Numeric	100	Count Upper threshold; if message count exceeds this value, incoming port is closed.
Count Upper Threshold	Numeric	1000	Count Lower threshold; if message count exceeded, reopens incoming port upon reaching lower threshold.

2.4.1.9 MSMQ Priority

Key	Valid value	Default	Description
Level	0-7	3	Default message priority (0-lowest .. 3 - normal .. 7 - highest).
Increase emergency priority	True / False	true	If set to `true`, emergency messages (from RAM) have 1 priority level above normal (logged) messages.

2.4.1.10 Unit Messages Watchdog 01 and 02

Key	Valid value	Default	Description
Enabled	True / False	True	Initial watchdog enable state.
Timeout (sec)	> 0 …	60	Timeout duration in seconds.
Action level	Predefined List [See description]	RESTART_APPLICATION	NONE: No action will take place. LOG_TO_FILE: Write to log file. LOG_TO_SCREEN: Write to screen [and log file]. RESTART_APPLICATION: Restart application.
Action delay (sec)	0 …	15	Duration, in seconds, from timeout to action execution.

Key	Valid value	Default	Description
Action enable abort	True / False	True	Enable the user to abort action, valid to RESTART_APPLICATION.

2.4.1.11 Server Address Cmd

Key	Valid value	Default	Description
Available Server IP.0			IP option #0
:	:	:	:
Available Server IP.9			IP option #9
Available Server Port.0			Port option #0
:	:	:	:
Available Server Port.9			Port option #9

Open **..\IniFile\GPRSManagerSetting.ini** file for editing and configure operational parameters according to the explanations provided in the following table.

Key	Valid value	Default	Description
TimeOut	Integer	8	Defines the timeout in seconds for the Linker to wait for an ACK from the unit.
Enable Auth Code	Integer	0	Enables the option to work in secure communication with the Cellocator units. If enabled and the unit is not configured, backward compatibility is used, normal behavior will continue.
Auth Table	String		Defines the authentication table used to authenticate the Linker for Cellocator units.
Save Sockets Log Period	Integer	0	Saves the status of each socket in the log for the configured amount of time in hours.
Max Transmission Delay	Integer	75	Re-checks the status of each unit in the configured amount of time. If the unit did not transmit any data in this period in minutes, the socket will be removed, and the unit will be marked as "disconnected".
Check TCP Connections	Integer	600	Runs the "TCP Connection Control" every defined number of seconds.
TCP Connections Control	Integer	75	Removes the connection from the unit's list if the unit did not transmit in the configured time in minutes.

Key	Valid value	Default	Description
Allow Multi Unit with Same IP	0/1	1	Some operators use NAT IP pools for the GPRS connections. In this case, the units will be shown under the same IP address. This flag allows the user to see only the latest unit that connected with this address (if set to 1) or to see all units with the same IP (if set to 0).
Save Ack Log	0/1	0	When set to '1' will save all ACK data sent by the Linker to the units. When set to '0', it will not save the data. Note that this log should be open ONLY for debug, since it will load the Linker once enabled.
Save Error Log	0/1	0	When set to '1' will save a specific error log with all the errors created by the GPRS Manager DLL library. When set to '0' it will not save this log.
Save Incoming Log	0/1	0	Saves every incoming message from TCP/UDP to a log file.
Save Connection Log	0/1	0	Saves every connection / disconnection / socket change to the error logs.
Unit Log Interval Minutes	Integer	-1	Creates a periodical log of all units' entries in the units' window, according to the interval defined. For each interval, a new log is created. Use '-1' to disable this function.
Statistics Log Interval Minutes	Integer	-1	Creates a periodical log of all units' statistics, according to the interval defined. Each row in the log contains Date/Time, DL Count, RX Count, New Units Count, No. Of Address Change, Number of Errors, Connect Count and Disconnect Count, relevant to the interval logged. For each interval, a new log is created. Use '-1' to disable this function.

2.4.2 *Installing and Configuring CM App*

2.4.2.1 Prerequisites

- Refer to the prerequisites in section 2.1.

2.4.2.2 Installing the CM App

To install the CM App, double-click the **CM App.exe** file and follow the onscreen instructions.

To verify the installation, ensure the following:

- CM App installation finished successfully without errors.
- CM App INI file is configured properly (see the following section for further details about configuring the INI file).

- Log folders created successfully according to the INI file.
- CM App can be launched successfully.

IMPORTANT: If you use the CSA protocol in your system, you must run CM App before you run the Correlator application. If you did not follow this order, you must restart the Correlator.

2.4.2.3 Configuring the CM App

Open <Install folder>\CSA\CM App.INI file for editing and configure operational parameters according to the explanations provided in the following table. Note that the parameters marked in blue must be changed; the rest can remain with the default values or can be changed according to your needs.

2.4.2.4 Communication

Key	Valid value	Default	Description
TCP Port	Number	236	TCP port following device port configuration, set -1 to disable TCP port.
UDP Port	Number	237	UDP port following device port configuration, set -1 to disable UDP port.
Send Immediate Ack	True/False	true	Defines whether the CM App will send Ack to units or not.

2.4.2.5 Logging

Key	Valid value	Default	Description
Path	String	C:\	Application log files folder.
Max File size (Byte)	Number	5000000[=5MB]	Max log file size, in Bytes.
ToGui	True / False	true	Displays the log onscreen.
ToFile	True / False	true	Enables writing to log file.
Active	True / False	true	Enables/disables all logging functions (GUI & File).
Flag_Exceptions	True / False	true	Logging flag
Flag_Communication Data (App-Device)	True / False	false	Logging flag
Flag_Pipe Data (App-App)	True / False	false	Logging flag
Flag_Units list management	True / False	false	Logging flag

Key	Valid value	Default	Description
Flag_Message parsing	True / False	true	Logging flag
Flag_General Information	True / False	true	Logging flag
Flag_Debug information	True / False	true	Logging flag
Flag_General Errors	True / False	true	Logging flag
Flag_Programming information	True / False	true	Logging flag
Unit Xml Log	True / False	false	Generates a debug log for each unit in XML format.

2.4.2.6 CommandControl

Key	Valid value	Default	Description
Wait For Respond seconds	Number	10	Max time interval for waiting for a response from unit.
Pending Timeout Sec	Number	20	The timeout of a total response session.
Send retries	Number	3	Max number of retries per request.

2.4.2.7 Pipe

Key	Valid value	Default	Description
CM Cmd Count	Number	0	CM App works with MSMQ (CM to CC Pipe should be empty as it is obsolete). This count should be set to 2 or more (depending on the number of Senders & Receivers).
CM Cmd 0	String	Cmd,NewMSMQ,.\Private$\cm_app_out,1-100000000,NXS02030.NXS02031,SENDER	After installation, the MSMQ for CSA UL message is set to .\Private$\cm_app_out Format example: Cmd,NewMSMQ,FormatName:DIRECT=TCP:207.232.46.122\Private$\out,**1-1000**,NXS02030.NXS02031,SENDER Where **1-1000** is the range of the unit IDs; make sure to set the full range of IDs.

Key	Valid value	Default	Description
CM Cmd 1	String	Cmd,NewMSMQ,.\Private$\cm_app_in,1-100000000,NXS02030.NXS02031,RECEIVER	After installation, the MSMQ for CSA DL message is set to .\Private$\cm_app_in Format example: Cmd,NewMSMQ,FormatName:DIRECT=TCP:207.232.46.131\Private$\in,**1-1000**,NXS02030.NXS02031,RECEIVER Where **1-1000** is the range of the unit IDs; make sure to set the full range of IDs.
Post Send Ack Flag	True / False	false	Used to add monitoring of CSA messages in the Communication Center – not used by Integration Tool.

2.4.3 *Installing and Configuring SMS Gateway*

2.4.3.1 Prerequisites

- Refer to the prerequisites in section 2.1.
- SMS broker application – see *Configuring Support for Bi-Directional SMS Communication*

2.4.3.2 Installing the SMS Gateway

To install the SMS Gateway, double-click the **SMSGateway.exe** file and follow the onscreen instructions.

To verify its installation, type this URL in the browser: **http://localhost/PSG/PSG.aspx**. The browser should show this message:

```
"False The following parameters are missing: Message PhoneNum
ApplicationType MsgDirection"
```

Note that SMS broker settings must be defined in the INI file, as described in the following sections.

2.4.3.3 Configuring the SMS Gateway

Configuration of the SMS Gateway application is done by modifying the PSG.ini file in the PSG folder under the Inetpub folder. Note that the parameters marked in blue in the following sections must be changed. All other parameters can remain with the default values or can be changed according to your needs.

2.4.3.4 General

The SMS Gateway configuration switches are listed in the table below.

Key	Valid value	Default	Description
isMaster	Boolean	True	Indicates if it is a master or slave installation.
isPointerWare	Boolean	True	Indicates if it uses the database for updating the SMS Billing Log and for reading SMS Providers configuration. Set to False.
isDBMultiChannel	Boolean	True	Indicates whether to use the unit's provider in the database to determine the channel to be used. Set to False.
isPrefixRoute	Boolean	False	SMS Providers and Channels will be used according to the INI configuration, see the following sections.

2.4.3.5 SMS Providers List

These options are used by the SMS Gateway if isPointerWare=false and\or isPrefixRoute=true (see the previous section).

Key	Valid value	Default	Description
SMS_Provider0	URL		Will be used to send Text\Unicode messages.
SMS_Provider0_Name	String		Provider name: ActiveSMS or NowSMS
SMS_Provider1	URL		Will be used to send ERM messages.
SMS_Provider1_Name	String		Provider name: ActiveSMS or NowSMS
SMS_Provider2	URL		Will be used to send Cellocator messages.
SMS_Provider2_Name	String		Provider name: ActiveSMS or NowSMS

2.4.3.6 SMS Providers' Channels List

These options are used by the SMS Gateway if isPrefixRoute=true (see the *General* section).

Key	Valid value	Default	Description
Provider_0_Channel_0_Prefix			This should not include the 3 digits of the international prefix and the + sign. Values can be separated with commas. Empty entries should be removed, or filled with a remark.
Provider_0_Channel_1_Prefix			
…			
Provider_1_Channel_0_Prefix			
Provider_1_Channel_1_Prefix			
…			

2.4.3.7 Queue

Key	Valid value	Default	Description
isEnabled	Boolean	True	If enabled the SMS Gateway will send uplink messages to MSMQ.
QPath	Full path in TCP format		The SMS Gateway will send uplink messages to this queue, which should be configured as the Correlator queue.

2.4.3.8 Logs

Key	Valid value	Default	Description
isEnabled	Boolean		If enabled, SMS Gateway will log its activity under **.\Logs**.

2.4.3.9 Database

The SMS Gateway will use these values when `isPointerware` = true, to connect to servers and to alleviate load on the billing log.

Key	Valid value	Default	Description
PointerDBServer	String	…	The DB to which the SMS Gateway will connect to if configured.
PointerDBUser	String	…	DB username.
PointerDBPwd	String	…	DB password.
isSMSLogsEnabled	String	…	Enable or disable the possibility to log sent and received SMSs.
SMSLogsDBServer	String		The DB to which the SMS Gateway will connect to store the SMS billing log.
SMSLogsDBUser	String		DB username.
SMSLogsDBPwd	String		DB password.

2.4.3.10 Cellocator

This table defines SMS Gateway configuration for Cellocator units.

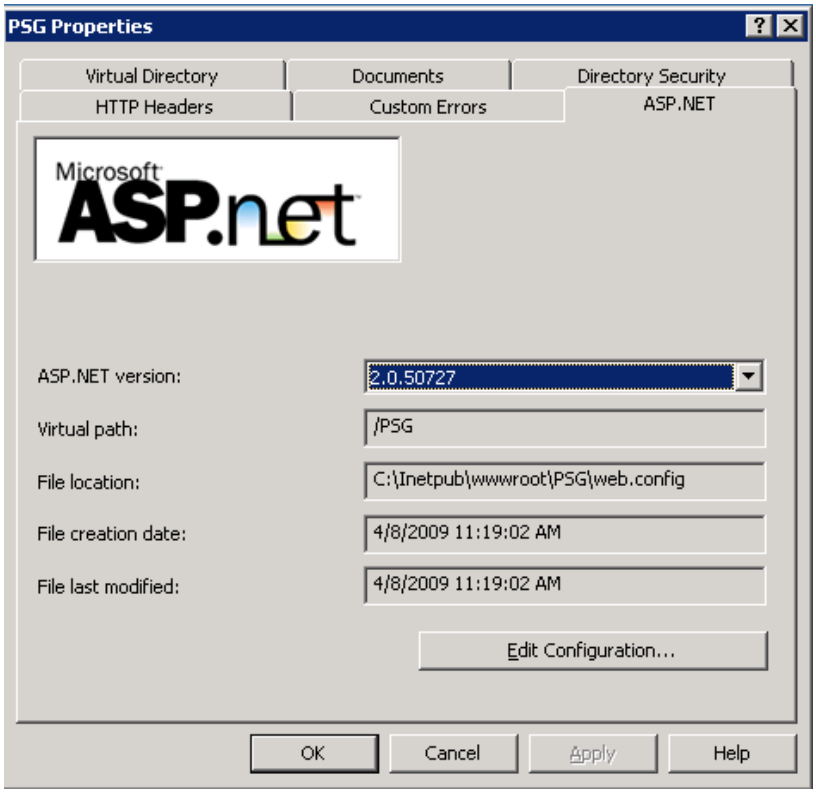
Key	Valid value	Default	Description
IsAuthenticate	String	…	Defines whether SMS sent to Cellocator unit is authenticated or not.
Authentication_Index_#	Table		Defines the authentication table to match the unit's authentication.

2.4.3.11 Additional Configuration Steps

1. Provide Read, Write and Modify permissions for user `IUSR_Servername`, for the following folders:
   - **wwwroot\PSG**

- **wwwroot\PSG\Logs**

2. Make sure the ASP.NET version is set to 2.0 for the SMS Gateway application:



3. Click **OK**, and then restart IIS.

2.4.4 *Installing and Configuring CorrelatorMax*

2.4.4.1 Prerequisites

- Refer to the prerequisites in section 2.1.
- MSMQ service.
- Server Regional Settings must be set to English (US).

2.4.4.2 Installing the CorrelatorMax

To install CorrelatorMax, double-click the **Correlator.exe** file and follow the onscreen instructions.

To verify the installation, ensure the following:

- CorrelatorMax installation finished successfully without errors.
- CorrelatorMax INI file is configured properly (see the following section for further details about configuring the INI file).
- Log folders created successfully according to the INI file.
- CorrelatorMax can be launched successfully (meaning the CorrelatorMax can establish a connection to the database and MSMQ successfully).

IMPORTANT: CorrelatorMax is a user space application and therefore user sessions cannot be logged off otherwise the application will be closed and no uplinks or downlinks will be processed by the application. We recommend you lock the user session in order to keep the application up.

In addition, if you use the CSA protocol in your system, you must run the CM App before you run the CorrelatorMax application. If you did not follow this order, you must restart the CorrelatorMax.

2.4.4.3 Configuring the CorrelatorMax

The following new and modified parameters should be configured in the CorrelatorMax INI file. Note that the parameters marked in blue must be changed; all other parameters can keep the default values or can be changed according to your needs.

2.4.4.4 General configuration

Key	Valid value	Default	Description
`CorrelatorAppId`	Numeric	0	When using multiple correlators and you need to distinguish between them, set different IDs for each Correlator. It will be logged in the Uplink table and when sending commands, each Correlator will retrieve commands that are sent to it directly. Can be set to 0, if all correlators should handle all commands.

Key	Valid value	Default	Description
CorrelatorAppName	String	Correlator Max IP	Defines the visual name of the Correlator in the application title.
Use_Fleet_Commands	True/False	false	Use_Fleet_ flags define the way Correlator handles data, `true` for Correlator fleet compatibility and `false` for Correlator IP compatibility. This flag controls commands format.
Use_Fleet_Coef	True/False	false	Set to false to display analog values with 1000 multiplier (1v = 1000).
Use_Fleet_SP	True/False	false	Set to false to support IP DB calls format.
Use_Fleet_Garmin_Handle	True/False	false	If set to true allows extended Garmin/MDT parsing (not suitable for IP DB).
Set_IP_Alarm_Trigger	True/False	true	If set to true will post msg type 0 as alarm.
Auto_Garmin_Reset	True/False	false	If set to true will send a reset command upon receival of F9 MDT command.
QInPath	String	.\Private$\CorrelatorMax	The Correlator Queue. The Correlator reads incoming messages from this queue. **NOTE**: The name of the Correlator IN Queue should match the name of the Linker OUT Queue. The Queue is created automatically by the installer in MSMQ.
QInternalPath	String		The Correlator internal queue. The Correlator writes a pending message (in memory) to this queue upon closing and reads it when restarting. **NOTE**: The name of the Correlator Internal Queue should be created manually in MSMQ.
SVR_db_Enable	Yes/no	Yes	DO NOT CHANGE THIS VALUE
Fleet_db_Enable	Yes/no	Yes	DO NOT CHANGE THIS VALUE
Safety_db_Enable	Yes/no	Yes	Disable if you do not have CSA events.
LogFile	Yes/no	Yes	Enables log files.
ErrorLog	Yes/no	Yes	Prints error to log.
Display	Yes/no	Yes	Enables display onscreen.

Key	Valid value	Default	Description
Trace_Debug	Yes/no	Yes	Prints debug info to log.
FilePath	String	.\Logs\	Logs file path.
Log_MaxSize	Integer	5	Logs max size in MB; a new file will be created when the file reaches this size.
DispalyRows	Integer	100	Number of rows to display onscreen.
dBFleetTarget_ThreadsNumber	Integer	5	Number of fleet insertion threads.
dBSafetyTarget_ThreadsNumber	Integer	1	Number of safety insertion threads.
TransparentInboundForwardData	Yes/no	No	Determines the way transparent msg type 5 are handled.
GpsPowerCheck	Yes/no	No	Determines whether to use IO status for GPS Power.
ContainersTTL	Integer	5	Cello Units Containers time to live in minutes. (Type 7)
Speed conversion ERM	Float	2.058 [Knot → kph]	A constant parameter applicable for ERM unit transmissions, used in order to convert vehicle speed GPS measurements from Knot into any other required velocity measurement units.
Speed conversion Cellocator	Float	1 [kph]	A constant parameter applicable for Cellocator unit transmissions, used in order to convert vehicle speed GPS measurements, from kilometers/hr into any other required velocity measurement units
NetworkTypeId	Integer	36	Internal application ID.
MSMQ_TimetoReceive	Integer	3	Read Timeout from MSMQ.
NetworkId	Integer	1	Instance ID, not in use.
DB_PAI_Request			If set to yes, calls spPAI_IsRawDataRequest to decide on SVR/Fleet DB.
CSASettingsFile	String	.\CorrelatorMax.ini	The new Correlator INI file includes the relevant CSA settings, so it points to itself.
TOP_Enable	Yes/no	No	If set to true/yes, TOP client will process TOP messages according to the TOP connection.

Key	Valid value	Default	Description
SMS_Reply	string	NO	Enables or disables DL messages by SMS channel, in case no reply comes from GPRS channel. If SMS Gateway is not installed, set to No.
UTC_Time_Check	String	NO	Enables or disables GPS time check in message in comparisons to the Gateway time and date.
Fleet_IP_Up	String	Yes	Determines whether the Correlator inserts the IP UP events into the DB.
Fleet_Only	Yes/no	Yes	DO NOT CHANGE, Insert data only to Fleet db.
Use_Ext_Input	Yes/no	No	Use_Ext_Input set parameters - @AnalogInput1, @AnalogInput2 to spCMUProcessUnitUplinkMsg.
Fleet_Always	Yes/no	Yes	Flag to define if all messages should be sent to fleet DB.
Alarms_To_Fleet	Yes/no	No	Defines if emergency message should be sent to fleet.
Limit_InternalQ	Yes/no	Yes	Defines if the internal queue size check is enabled.
InternalQ_Size	Numeric	200	Defines the internal queue message count threshold.
Restore_InternalQ	Yes/no	Yes	Flag to read internal queue stored in MSMQ upon startup.
LoadConversionDll	Yes/no	No	Valid only for Fleet Management with UTM coordinates type using conversion DLL.
GPSValidityCheck	Yes/no	Yes	Set to no if you would like to get GPS coordinates when validation fails (wrong GPS mode).
Type0_EventData_Using_XML	Yes/no	Yes	Writes type 0 event data information as XML.
Type0_EventData_Using_SP	Yes/no	No	Writes type 0 event data information using fleet DB specific tables.
CM_Enable	Yes/no	No	Enables CSA protocol – only if there is a CM App connected.

2.4.4.5 Insertion Type

Key	Valid value	Default	Description
28.Type11_EventData_Using_XML	Yes/no	Yes	Write type 11 module 28 "General Status Event" data as XML in ModuleData.
28.Type11_EventData_Using_SP	Yes/no	No	Write type 11 module 28 "General Status Event" data using fleet DB specific tables.
40.Type11_EventData_Using_XML	Yes/no	Yes	Write type 11 module 40 "Measurement Readings" data as XML in ModuleData.
40.Type11_EventData_Using_SP	Yes/no	No	Write type 11 module 40 "Measurement Readings" data using fleet DB specific tables.
42.Type11_EventData_Using_XML	Yes/no	Yes	Write type 11 module 42 "Nano Inherent Sensors" data as XML in ModuleData.
42.Type11_EventData_Using_SP	Yes/no	No	Write type 11 module 42 "Nano Inherent Sensors" data using fleet DB specific tables.
44.Type11_EventData_Using_XML	Yes/no	Yes	Write type 11 module 44 "MultiSense Additional Information" data as XML in ModuleData.
44.Type11_EventData_Using_SP	Yes/no	No	Write type 11 module 44 "MultiSense Additional Information" data using fleet DB specific tables.
45.Type11_EventData_Using_XML	Yes/no	Yes	Write type 11 module 45 "Full System MultiSense Readings" data as XML in ModuleData.
45.Type11_EventData_Using_SP	Yes/no	No	Write type 11 module 45 "Full System MultiSense Readings" data using fleet DB specific tables.

2.4.4.6 Total number of GPRS servers

Key	Valid value	Default	Description
ERMServers_Number	1…N	0	The number of running Cellnet servers within Pointer's Cellular gateway. Set to 0.
CellocatorServers_Number	1…M	1	The number of running Cellocator GPRS servers (Cellnet Linker) within Pointer's Cellular gateway.

2.4.4.7 GPRS Cellocator servers

Key	Valid value	Default	Description
1.QoutPath	Full path in TCP format	.\private$\Linker	FormatName:DIRECT=TCP:<ip address>\private$\<queue name> **NOTE**: The name of the Correlator Out Queue should match the name of the Linker IN Queue. The Queue is created automatically in MSMQ by the installer.
:			:
:			:
N. QoutPath			FormatName:DIRECT=TCP:<ip address>\private$\<queue name>

2.4.4.8 GPRS ERM servers

Key	Valid value	Default	Description
1.QoutPath	Full path in TCP format	.\private $\Linker	FormatName:DIRECT=TCP:<ip address>\private$\<queue name>
:			:
:			:
N. QoutPath			FormatName:DIRECT=TCP:<ip address>\private$\<queue name>

Note the following:

- Only the first X Queue path entries as indicated by CellocatorServers_Number/GPRS Cellocator Servers are referenced by the CorrelatorMax application. Declared but not running GPRS servers may cause messages to get lost in the system. If declared queues do not exist upon service start-up, an error will be returned.
- The HW type section in the INI file describes the different Cellocator HW types and the conversion rate for voltage and IO for each of the HW types. These settings should not be changed.

2.4.4.9 GSM Server and Ringer

GSM server will be used for sending SMS messages; the ringer will be used for waking up ERM units.

Key	Valid value	Default	Description
Http_Page	String	http://localhost/ PSG/PSG.aspx	Determines the HTTP page to which the server will access when trying to send an SMS to a unit. If you are using SMS Gateway, set the SMS Gateway URL.
QoutRingPath	String	.\private$\Linker	Defines the Queue path which will connect to the ringer and will activate

Key	Valid value	Default	Description
			the ringer to wake up ERM units. Note that even when the ringer is not in use, the queue should be defined in MSMQ and set properly in the INI file. **NOTE**: The name of the Ringer Queue can match the Linker IN Queue. There is no need to define a special queue for the ringer.

2.4.4.10 DB configuration

Key	Valid value	Default	Description
DbDL_delay_SVR	Integer	1	Defines the downlink delay to the SVR DB.
DSN_name_SVR	String		Defines the name of the DB instance.
Db_name_SVR	String	TWPQueues	Defines the name of the SVR DB in the instance.
User_name_SVR	String		Defines the name of the SVR DB user.
Password_SVR	String		Defines the name of the SVR DB user's password.
Timeout_SVR	Integer	30	
DbDL_delay_Fleet	Integer	1	Defines the downlink delay to the Fleet DB.
DSN_name_Fleet	String		Defines the name of the DB instance.
Db_name_Fleet	String	TWPQueues	Defines the name of the Fleet DB in the instance.
User_name_Fleet	String		Defines the name of the Fleet DB user.
Password_Fleet	String		Defines the name of the Fleet DB user's password.
Timeout_Fleet	Integer	30	
DbDL_delay_Safety	Integer	1	Defines the downlink delay to the Fleet DB.
DSN_name_Safety	String		Defines the name of the DB instance.
Db_name_Safety	String	TWPQueues	Defines the name of the Fleet DB in the instance.
User_name_Safety	String		Defines the name of the Safety DB user.
Password_Safety	String		Defines the name of the Safety DB user's password.
Timeout_Safety	Integer	30	

2.4.4.11 Logging

The following table defines CSA message controller logging options.

Key	Valid value	Default	Description
Path	String	C:\Program Files (x86)\Pointer\CorrelatorMax\Logs\	Application log files folder.
Max File size (Byte)	Number	5000000[=5MB]	Max log file size, in Bytes.
ToGui	True / False	false	Displays the log onscreen.
ToFile	True / False	false	Enables writing to a log file.
Active	True / False	false	Enable/disable all logging functions (GUI & File).
Flag_Exceptions	True / False	True	Logging flag
Flag_Communication Data (App-Device)	True / False	True	Logging flag
Flag_Pipe Data (App-App)	True / False	True	Logging flag
Flag_Units list management	True / False	True	Logging flag
Flag_Message parsing	True / False	true	Logging flag
Flag_General Information	True / False	true	Logging flag
Flag_Debug information	True / False	true	Logging flag
Flag_General Errors	True / False	true	Logging flag
Flag_Programming information	True / False	true	Logging flag
Unit Xml Log	True / False	false	Generates a debug log for each unit in XML format.

2.4.4.12 CommandControl

Key	Valid value	Default	Description
Wait For Respond seconds	Number	10	Max time interval for waiting for a response from the unit.
Pending Timeout Sec	Number	20	The timeout of a total response session.
Send retries	Number	3	Max number of retries per request.

2.4.4.13 MSMQ

Key	Valid value	Default	Description
Count	Number	2	For the CM App to work with MSMQ, count should be set to 2 or more (depends on the number of Senders).
0	String	Receiver,.\Private $\cm_app_out	This parameter is set automatically by the installer. Change the server IP, make sure to define the queue manually. The entry format is as follows: *Receiver,FormatName:DIRECT=TCP:207.232 .46.131\Private$\out*
1	String	Sender,.\Private$\ cm_app_in	This parameter is set automatically by the installer. Change the server IP, make sure to define the queue manually. The entry format is as follows: *Sender,FormatName:DIRECT=TCP:207.232. 46.131\Private$\in*

2.4.4.14 Coordinate's conversion

Conversion is applicable only for the Fleet Management service. For the SVR service, coordinates are always held in the database as special Pointer integer representations of geographic coordinates.

Key	Valid value	Default	Description
LoadConversionDll	Yes/No	no	Determines whether to load and execute geographic coordinates conversion to comply with any UTM based maps. Shall be set to YES only if Fleet Management site is connected to UTM based maps.
CoordinatesType	UTM/GEO/ALL/ BOTH	ALL	
GeographicType	From Table	4034	This parameter denotes the geographical conversion constant per region. Consult Pointer Customer Support if you don't know the exact value.
ProjectedType	From Table	28193	This parameter denotes the projection conversion constant per region. Consult Pointer Customer Support if you do not know the exact value.
Offset_X/ Offset_Y	Integer [Meters]	50000 / 500000	The offset to be added to the X/Y dimensions after the conversion.

Key	Valid value	Default	Description
Sign_X/ Sign_Y	1/-1	1 / -1	A sign multiplication to the results of the conversion, after the offset addition (multiply X/Y in +/-1).

2.4.4.15 HW Types

The CorrelatorMax distinguishes between different HW types, external power and battery voltage, IO configuration and external power and battery thresholds.

Key	Valid value	Default	Description
SupplyVoltage	Float		A multiplier parameter to multiply the value received from the unit to provide the real vehicle supplied voltage.
BackupBattery	Float		A multiplier parameter to multiply the value received from the unit to provide the real backup battery supplied voltage.
OptionalAnalog	Float		A multiplier parameter to multiply the value received from the unit to provide the real analog input reading from the unit.
IOStatus	Integer		Multiplier to mask IO status – for future usage.
ExtPW_Treshold	Float		External power lowest threshold. If the power received is lower, it will be registered as 0 in the database.
IntPW_Treshold	Float		Backup battery power lowest threshold. If the power received is lower, it will be registered as 0 in the database.

2.4.5 *Installing and Configuring Database Objects/Server*

2.4.5.1 Prerequisites

- Refer to the prerequisites in section 2.1.
- Windows OS – English only
- Microsoft SQL Server 2000/2008/2008R2/2012 and above – English only
- SQL Server Authentication: Mixed mode (SQL Server and Windows)

2.4.5.2 Verifying the installation

Please refer to section 2.3.

Once the batch file execution has terminated one can verify the following steps:

- Verify the creation of the database according to the type selected (TWPQueues or CellocatorHub)

- Verify all database tables were created:
  - CMUDLQueue
  - UplinkMsgLog
  - SMSBillingLog
  - SMSTransport
  - ModularLog
- Verify all database stored procedures were created:
  - spCMUDLQueue_Get
  - spCMUProcessUnitUplinkMsg
  - spCMUs_Get
  - spInsertSMSBillingLog
  - spPAI_IsRawDataRequest

2.5 Configuring Support for Bi-Directional SMS Communication

The SMS Gateway supports bi-directional SMS communication with Cellocator units; the SMSC communication is done by 3rd party applications that have an integrated interface with the SMS Gateway via HTTP requests.

SMS communication can be either a direct communication with the GSM provider (the protocol is called SMPP) or by using a GSM modem with SIM card (both options require a 3rd party application).

Currently the SMS Gateway supports two SMS broker applications:

- **NowSMS** (www.nowsms.com): Supports both direct communication via SMPP and indirectly via GSM modem. Verify with your GSM provider if the SMPP protocol is available for use.
- **ActiveSMS** (http://www.intellisoftware.co.uk/products/activesms/): Supports communication via one or more GSM modems.

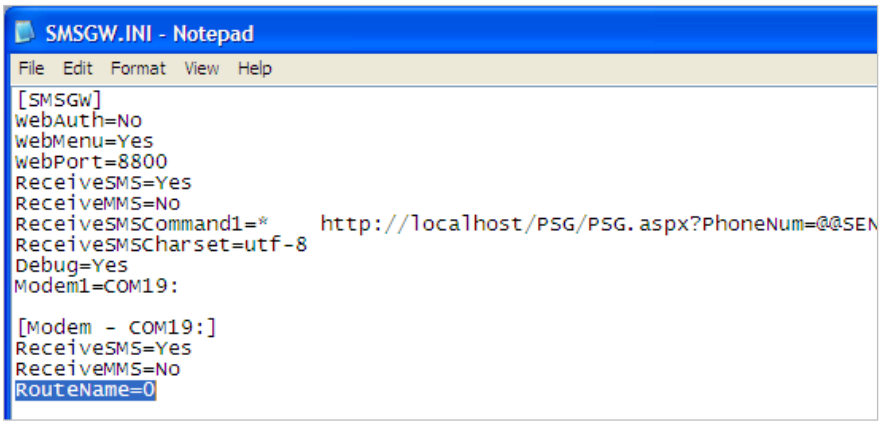
The configuration steps in the following sections refer ONLY to the bi-directional communication and should be applied AFTER installing and configuring the SMS broker application with the SMS modem.

Configuration includes **SMS Downlinks**, which are redirected from the SMS Gateway to the SMS Broker application and from there to the Cellocator unit and **SMS Uplinks**, which are sent from Cellocator unit to the SMS modem and are received by the SMS Broker that redirects them to the SMS Gateway.

2.5.1 *Configuring an SMS Gateway and NowSMS Interface*

2.5.1.1 Configuring Downlinks

1. Open the **PSG.INI** file under the PSG (Pointer SMS Gateway) folder, and in the `SMS_Provider2` entry, set the NowSMS URL. The URL should include the default port used by NowSMS; assuming NowSMS is installed on the PSG (Pointer SMS Gateway) server, use this URL: **http://localhost:8800/**
2. Open the **SMSGW.INI** file under the NowSMS folder, and under the `Modem` section added after configuring the SMS Modem or SMSC provider, add this line: `RouteName=0`

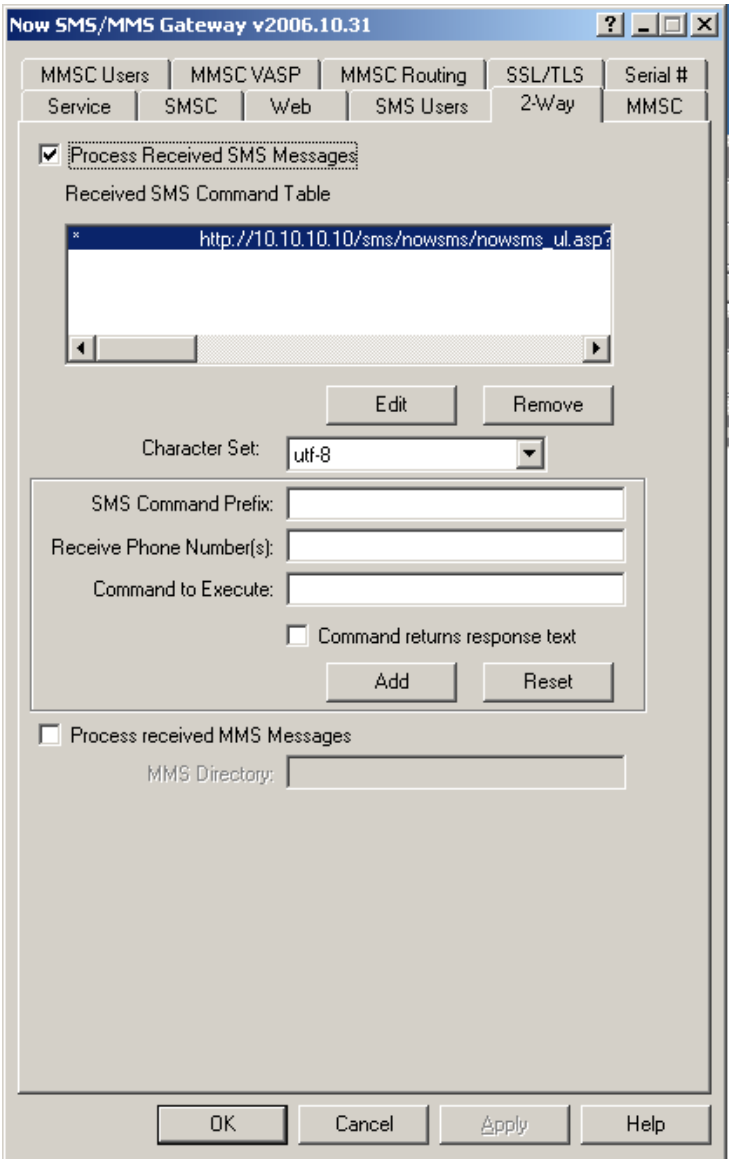


3. Save the file.

2.5.1.2 Configuring Uplinks

1. Open the NowSMS console, and in the 2-Way tab define the following:
   - Select the **Process Received SMS Messages** checkbox
   - Add * in the **SMS Command Prefix** text box.
   - Ensure the **Receive Phone Number(s)** text box is left empty.
   - Add this URL in "Commands to Execute", (if needed, change **localhost** to the IP address of the SMS Gateway server):

     http://localhost/PSG/PSG.aspx?PhoneNum=@@SENDER@@&Message=@@FULLSMS@@&Binary=@@BINARY@@&Data=@@DATA@@&MsgDirection=0&SMSProvider=0&CellProvider=@@SMSCROUTE@@&CallBackNumber=@@SENDER@@&ApplicationType=38



2. Click **OK** to complete the uplinks configuration.

2.5.2 *Configuring an SMS Gateway and ActiveSMS Interface*

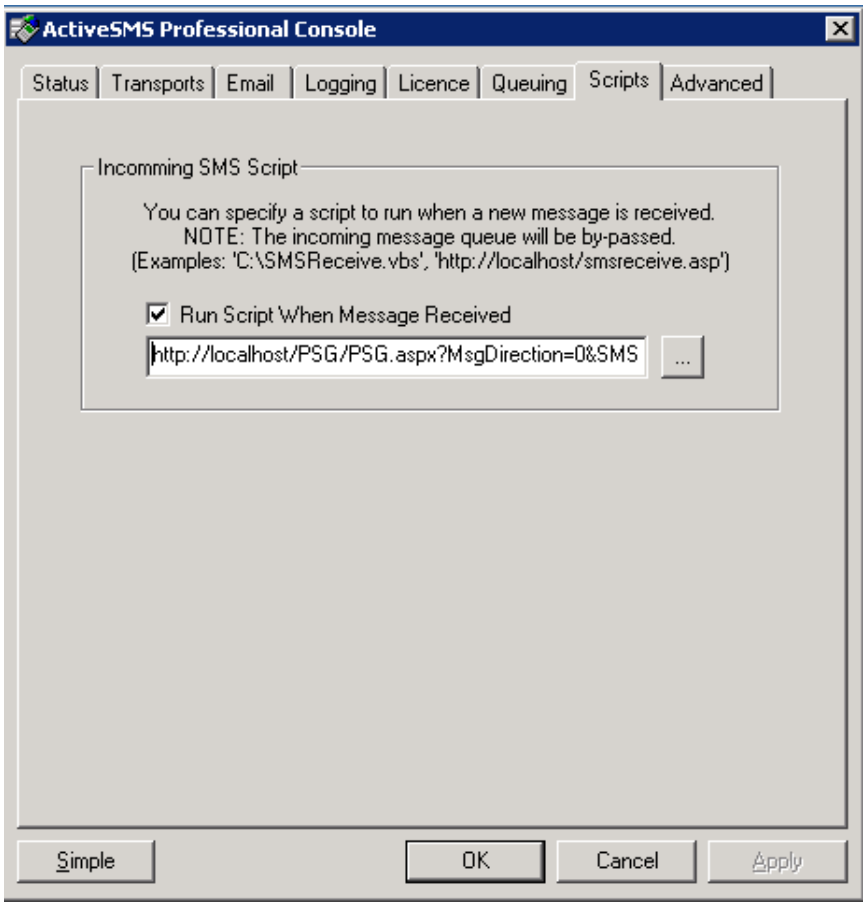
Note that if ActiveSMS is installed on a different server than the SMS Gateway, you will need to install a SMS Gateway Slave on that server to allow the Master SMS Gateway to communicate with the remote ActiveSMS server.

2.5.2.1 Configuring Downlinks

Open the **PSG.INI** file under the PSG folder (Pointer SMS Gateway), and in the `SMS_Provider2` entry, set the ActiveSMS URL. The URL should include only the server IP; assuming ActiveSMS is installed on the SMS Gateway server, use this URL:
http://localhost/

2.5.2.2 Configuring Uplinks

1. Open the ActiveSMS console, and in the **Scripts** tab define the following:
   - Select the **Run Script When Message Received** checkbox
   - Add this URL in the text box below (if needed change **localhost** to the IP address of the SMS Gateway server):
     http://localhost/PSG/PSG.aspx?MsgDirection=0&SMSProvider=0&ApplicationType=38



2. Click **OK** to complete the uplinks configuration.

2.6 Installing MSMQ and IIS

The following procedure describes how to install IIS and MSMQ, which are both part of the Windows OS Infrastructure.

➤ **To install MSMQ and IIS**

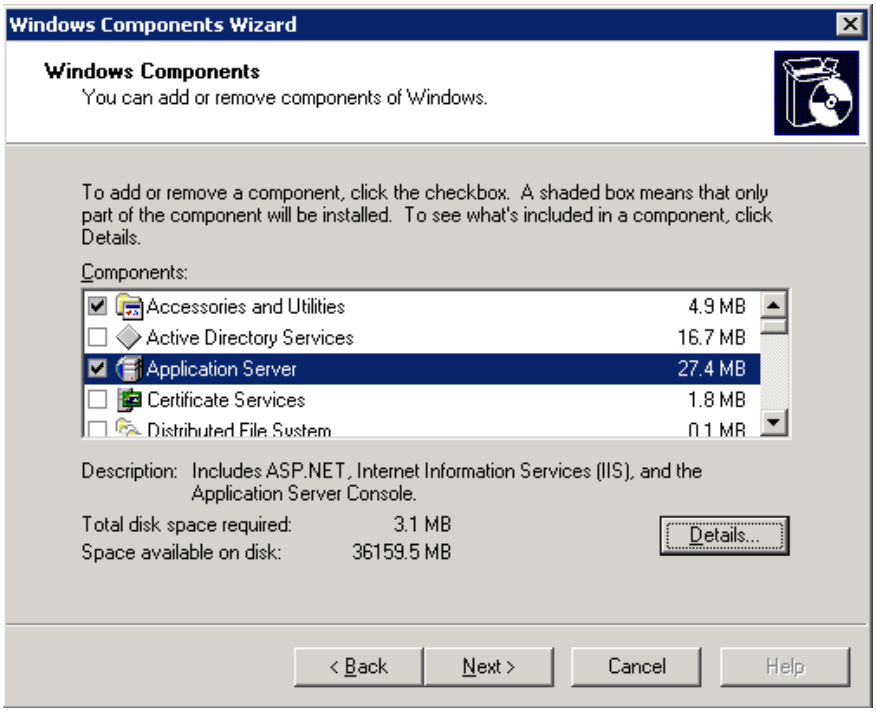
NOTE: The instructions below are suitable for Windows 2003 Server; there may be minor differences in different Windows OS versions.

If the IIS Component is installed AFTER installing .NET Framework, you should run the following in order to register the IIS under the ASP.NET environment:

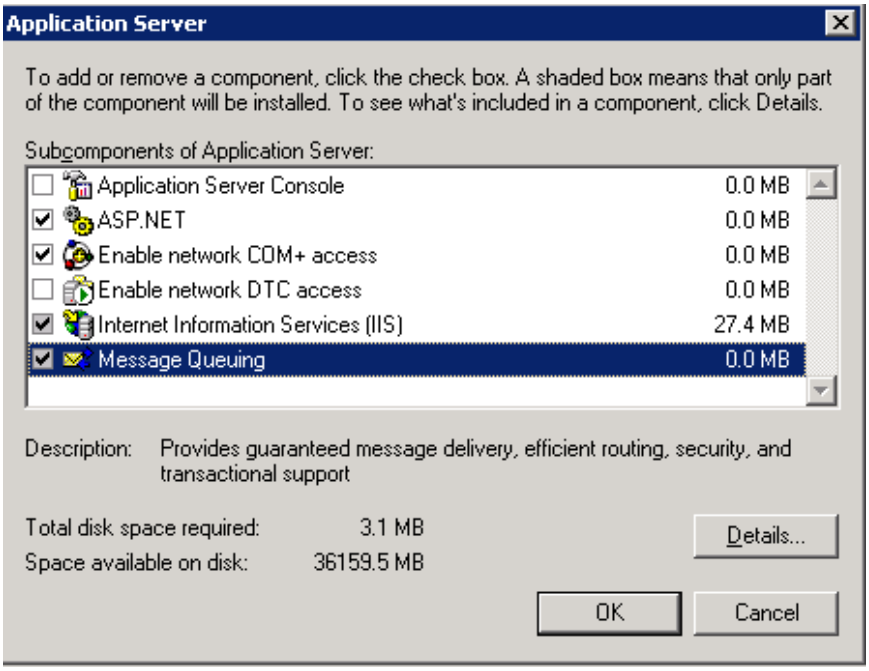
```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_regiis.exe -i
```

---

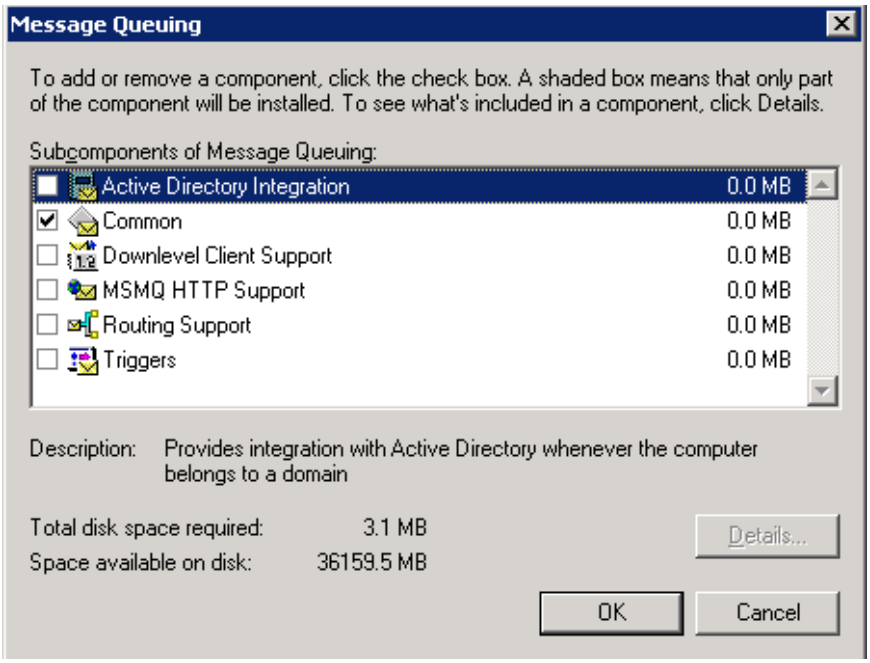
1. Go to **Control Panel>Add or Remove Programs>Add/Remove Windows Components**. In the displayed Windows Components dialog, select **Application Server** and click **Details**.



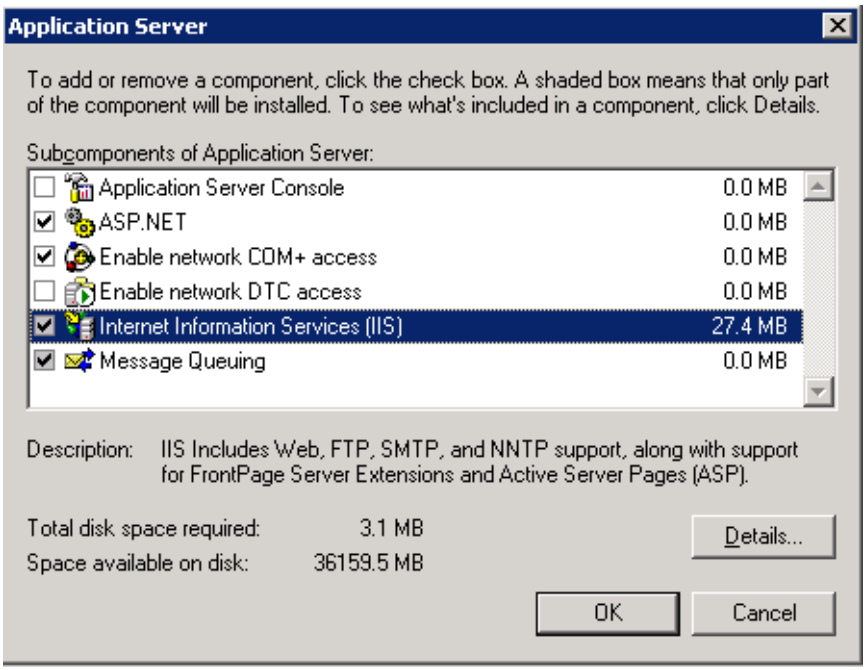
2. In the Application Server dialog, select **Message Queuing** and click **Details**.



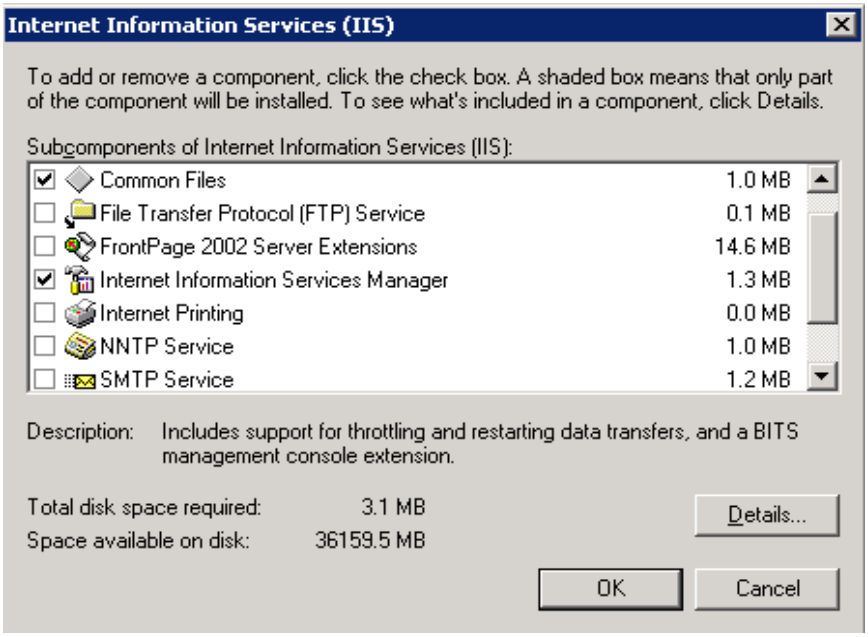
3. In the Message Queuing dialog, select the **Common** component and click **OK** to go back to the Application Server dialog.



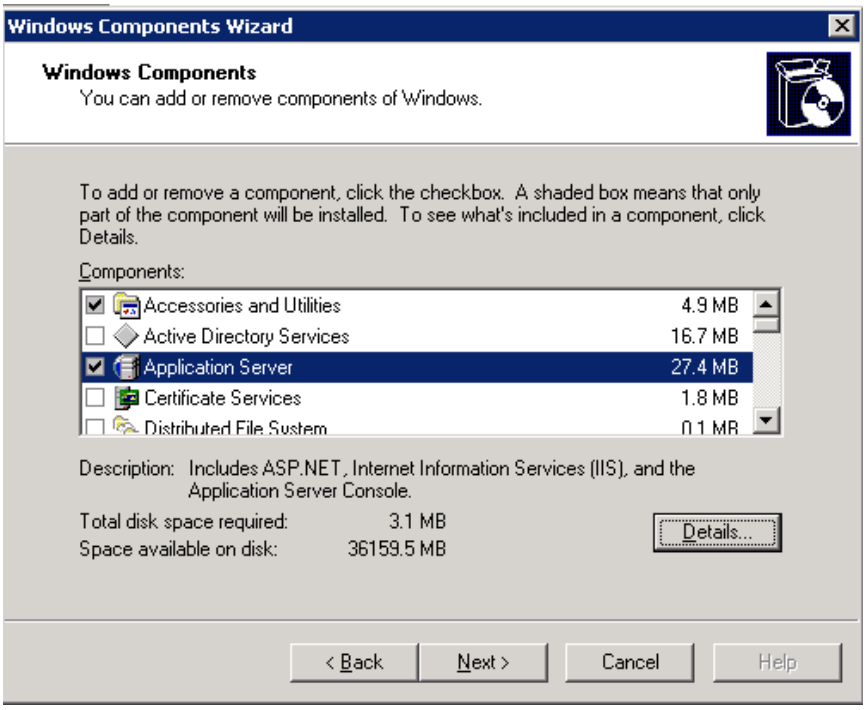
4. In the Application Server dialog, select **IIS** and click **Details**.



5. In the IIS dialog, select the **Common Files** and **IIS Manager** components (the 'World Wide Web Service' will be selected automatically) and click **OK** to go back to the Application Server dialog.



6. In the Application Server dialog click **OK** to go back to the Windows Components Wizard. Click **Next** in the Wizard dialog and when asked, browse to the Windows Installation Disk or to the I386 boot folder, to locate the installation files needed.



2.7 Advanced Configuration (optional)

This section includes advanced configuration steps, primarily focused on the I/O table mapping.

2.7.1 *I/O table mapping*

- Each I/O has a mapping in the CorrelatorMax.ini.
- The parameter is referred by Name = *ByteOffset*, *BitOffset*, *Inverted*
  - ByteOffset in I/O bytes- 0-3
  - BitOffset in byte (0-7)
  - Inverted, values: Normal = 0 or Inverted = 1
- If the parameter is not defined, a value of null will be set to the parameter.

The I/O Parameters are listed below:

- DrivingStatus
- IgnitionPhysical
- IgnitionAccelerometer
- Door
- Shock
- Distress
- Lock
- Unlock
- TamperSwitch
- GP1
- GP2
- Movement
- UsbPowerConnected
- PackageIsOpen
- Button1
- Button2
- Arm
- Disarm
- Siren
- GradStop
- GPSPower
- LedOut
- D8DTCConnected
- Blinkers

- ExternalPower
- StandardImmobilizer
- ChargerStatus
- CFEInBitmap
  - Only if CFE is enabled (Byte 24 bits 2-4 is greater 2 to 7)
  - Value is a Bitmap, CFE0 is in Bit0.. CFE5 is in Bit5.
- CFEOutBitmap

2.7.2 *Launching the Process Monitor Service on a VM*

When installing the Integration Tool, as described on page 15, you can choose to launch the Process Monitor Service upon completion of the installation process, or run it manually at a later date (via the Windows Task Manager). If you selected to launch the Process Monitor Service, the CorrelatorMax, Cellnet Linker, and CM App components are launched automatically on a local computer.

However, on a VM, the Process Monitor Service will not appear to launch, and the CorrelatorMax, Cellnet Linker, and CM App also do not launch. This is because you cannot see processes running under different session IDs (local sessions are typically Session 0, while sessions on a VM are Session 1; these session IDs can be viewed when you access the Windows Task Manager on the VM).

WORKAROUND

To launch the Process Monitor Service in your current VM session, first stop the Process Monitor Service in the Services tab of the Windows Task Manager (right-click on the service and select Stop). Then stop the process in the Details tab of the Task Manager (right-click and select End task).

Then restart the Process Monitor Service in your VM session.

3 Integration

This chapter describes how to use the Integration Tool to integrate commands and uplinks with your database, and includes the following sections:

- **Getting Started with the Cellocator Integration Tool**, see below.
- **How to Receive Messages Sent from the Unit to the Database**, page 56
- **How to send Commands from the Server to the Unit**, page 108
- **Advanced Integration: Message Parsing**, page 67

3.1 Getting Started with the Cellocator Integration Tool

This section describes how to get started with the Integration Tool, as described in the following steps.

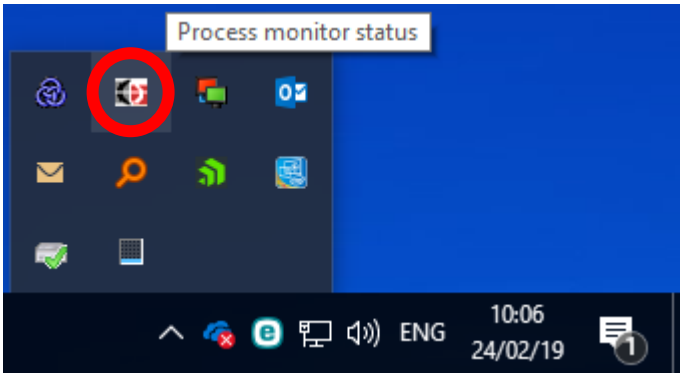
1. Complete the installation, as described in *Installation*.
2. Make sure the installed components are up and running, by performing one of the following:
   - Access the Process Monitor Status notification icon to verify the status of each component. This icon can also be accessed in a remote session; see the following *Reviewing the status of Integration Tool components* section.
   - Verify the components are installed by following the instructions on section 2.3
   - Launch the relevant components from the Windows *Start* menu.
3. Start sending messages from the unit to the database, as described on page 56, or sending commands from the database to the unit, as described on page 108.

3.1.1 *Reviewing the status of Integration Tool components*

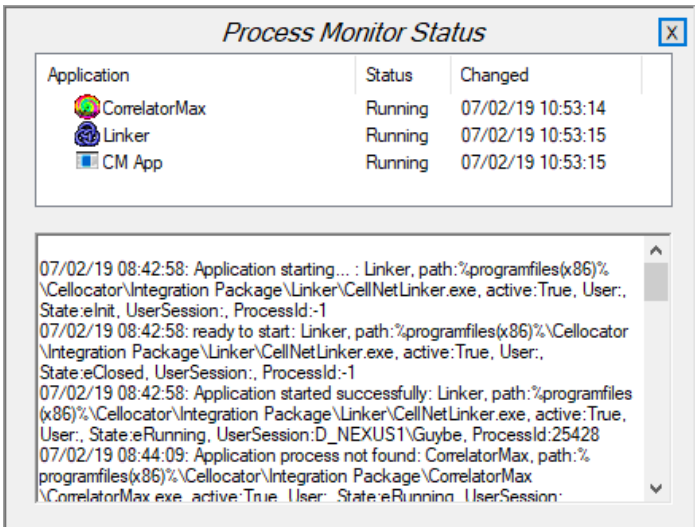
This section describes how you can use the Process Monitor Status notification icon to verify the status of each Integration Tool component.

➤ **To review the status of Integration Tool components**

1. From the Windows notification area, click on the Process Monitor Status notification icon, as shown below.



2. In the displayed *Process Monitor Status* window, you can view the status of the installed applications, as shown below. The upper pane shows the status of the application and a record of the last change to that status, while the lower pane shows a text log.



Note that if the Process Monitor Service is currently inactive, this window is highlighted in red, and the notification icon will also display an alert.

3. Perform any of the following actions by right-clicking on the relevant application and selecting the relevant menu option:

   - **Open Logs Folder:** Opens Windows File Explorer in the **Logs** folder of the application. The last modified file is selected by default.
   - **Open Application Folder:** Opens Windows File Explorer in the application executable folder.
   - **Monitor:** Click to select for monitoring; if not selected, the application is not monitored and can be closed.
   - **Kill Application Process:** Kills the process (in the same way when right-clicking the process in the *Task Manager* window to end the task). If monitoring is enabled, the Process Monitor Service application will restart automatically.

     Note that the application may be unavailable if you are running another session; however, the *Process Monitor Status* window displays and controls applications running on *all* sessions.

3.2 How to Receive Messages Sent from the Unit to the Database

The interface to the Integration Tool is the database; messages are managed using your preferred SQL management tool (such as SQL Server Management Studio Tool or MySQL Workbench).

The interface can be via the level of the tables in the database or via changes in the levels of the stored procedures executed by the CorrelatorMax application. The relevant fields are described in the following sections and tables.

Note that there are two actual methods of receiving the messages:

- The new message format introduced in version 2.7.100 (located under **CellocatorHub** in the *Databases* folder and explained below).
- The legacy method (located under **TWPQueues** under the same *Databases* folder and explained on page 57).

3.2.1 *Received Messages using CellocatorHub and TWPQueues Tables*

This section details the SQL scripts that generate the tables; below are scripts used in Microsoft SQL.

For CellocatorHub the script used is as follows:

```
SELECT TOP (1000) [CMUId] ,[Lat] ,[Long] ,[Alt] ,[HWID] ,[FWVer] ,[Protocol]
,[MessageType] ,[MessageInitiative] ,[MessageSource] ,[UnitMode] ,[PLMN] ,[Hibernation]
,[Analog1] ,[Analog2] ,[Analog3] ,[Analog4] ,[Odometer] ,[DriverID] ,[Sim]
,[MultiPurpose] ,[GPSDateTime] ,[LastGPSFix] ,[GPSMode1] ,[GPSMode2] ,[GPSSpeed]
,[GPSCourse] ,[NumOfSat] ,[UTCTime] ,[GWTime] ,[InsertDate] ,[IP] ,[TXIndex]
,[NetworkTypeId] ,[TRId] ,[TRSpecificData] ,[CorrelatorAppId] ,[LinkerAppId]
,[HRNetwork] ,[GPSCommStatus] ,[Numerator] ,[RawData] ,[TotalIO] ,[DrivingStatus]
,[IgnitionPhysical] ,[IgnitionAccelerometer] ,[Door] ,[Shock] ,[Distress] ,[Lock]
,[Unlock] ,[TamperSwitch] ,[GP1] ,[GP2] ,[Movement] ,[UsbPowerConnected]
,[PackageIsOpen] ,[Button1] ,[Button2] ,[Arm] ,[Disarm] ,[Siren] ,[GradStop]
,[GPSPower] ,[LedOut] ,[D8DTCConnected] ,[Blinkers] ,[ExternalPower]
,[StandardImmobilizer] ,[ChargerStatus] ,[CFEInBitmap] ,[CFEOutBitmap] ,[TripId]
,[ManeuverID] ,[ManeuverUsage] ,[AccidentBuffer] ,[ModularData] ,[Modules]  FROM
[CellocatorHub].[dbo].[UplinkMsgLog]
```

For TWPQueues the script used is as follows:

```
SELECT TOP (1000) [RMUId] ,[CellX] ,[CellY] ,[CellDateTime] ,[GPSX] ,[GPSY]
,[GPSDateTime] ,[Speed] ,[Direction] ,[NumOfSat] ,[LocQuality] ,[Data] ,[Address]
,[EngineOn] ,[ExtInputA] ,[ExtInputB] ,[ExtInputC] ,[ExtInputD] ,[ExtInputE]
,[ExtInputF] ,[VersionNum] ,[IMSI] ,[IP] ,[BytesTotal] ,[TXIndex] ,[MainArea]
,[MainAnt] ,[MainDBMQuality] ,[InputVoltage] ,[BackBatVoltage] ,[GPSPDOP] ,[GPSHDOP]
,[GPSVDOP] ,[GPSHEIGHT] ,[NetworkTypeId] ,[UnitMsgId] ,[InsertDate] ,[UTCTime]
,[DriverId] ,[TxReasonId] ,[AlertReason] ,[Hibernation] ,[HRNetwork] ,[UnitMode]
,[MileCounter] ,[GPSMode1] ,[GPSMode2] ,[OutputA] ,[OutputB] ,[OutputC] ,[OutputD]
,[OptionalInput] ,[GPSCommStatus] ,[RawData] ,[PLMN] ,[TotalIO] ,[IO_Byte1_bit2]
,[IO_Byte1_bit3] ,[IO_Byte1_bit4] ,[IO_Byte2_bit0] ,[IO_Byte2_bit1] ,[IO_Byte2_bit3]
,[IO_Byte2_bit4] ,[IO_Byte2_bit5] ,[IO_Byte2_bit6] ,[IO_Byte2_bit7] ,[SN]
,[MessageType] ,[UnitData] ,[MsgProtocol] ,[TripId] ,[ManeuverID] ,[ManeuverUsage]
,[AccidentBuffer] ,[ItemId] ,[CorrelatorAppId]  FROM [TWPQueues].[dbo].[UplinkMsgLog]
```

3.2.2 *Receiving Messages via TWPQueues Tables*

3.2.2.1 Uplink Message Log Interface

Field Name	Field Description	Type	Example
RMUId	Cellocator unit's ID as appears on unit exterior.	Integer	203729
CellX	Please ignore (always 0).	Integer	
CellY	Please ignore (always 0).	Integer	
CellDateTime	Date & Time when an uplink message was received by CorrelatorMax.	DateTime	'2009-06-21 11:20:15.720'
GPSX	Longitude coordinate of current position fix (divide by 600000 to get the value in degrees).	Integer	20981126
GPSY	Latitude coordinate of current position fix (divide by 600000 to get value in degrees).	Integer	19264606
GPSDateTime	Unit GPS time NULL will be inserted when the message doesn't include GPS data.	DateTime	'2009-06-21 11:20:15.720'
Speed	Current speed (absolute value of the vector) in km/hour.	Integer	67
Direction	Direction (angle) of the speed vector.	Integer	328
NumOfSat	Number of satellite measurements used for current position fix.	Integer	11
LocQuality	GPSPDOP, see Cellocator protocol.	Integer	107
Data	Used by SVR system. Please ignore.	Char(16)	
Address	Please ignore (always empty).	Nvarchar(255)	
EngineOn	Ignition status (1 - On, 0 - Off)	Integer	1
ExtInputA	Input lock In (IO byte2, bit 2)	Integer	0,1, NULL
ExtInputB	Input Unlock In (IO byte1, bit 7)	Integer	0,1, NULL
ExtInputC	Input Ignition On (IO byte1, bit 5)	Integer	0,1, NULL
ExtInputD	Input door (IO byte1, bit 0)	Integer	0,1, NULL
ExtInputE	Input Blinkers In (IO byte1, bit 1)	Integer	0,1, NULL
ExtInputF	Input Distress (IO byte1, bit 6)	Integer	0,1, NULL
VersionNum	HW Type and FW version in this format: "HW" + HW type (New HW Type as it appears in Wireless protocol) + "SW + SWtype +"." + SW Variant.	VARCHAR[25]	HW<222> SW<28.2>

IMSI	SIM card identifier received from legacy units. Please ignore.	Char[20]	
IP	Unit's IP address (available only in IP-UP event).	Char[15]	
BytesTotal	Used by SVR system. Please ignore.	Integer	
TXIndex	Message index in concatenated messages received from legacy units. Please ignore.	Integer	
MainArea	Received from legacy units. Please ignore.	Integer	
MainAnt	Received from legacy units. Please ignore.	Integer	
MainDBMQuality	Received from legacy units. Please ignore.	Integer	
InputVoltage	Main Power (Analog input 1) in 1000V, rounded down in 0.25V units. Example: 13.3V->13000	Integer	12500
BackBatVoltage	Backup Battery (Analog input 2) in 1000V, rounded down in 0.25V units. Example: 0.57V->500	Integer	4250
GPSPDOP	GPS PDOP value (Position Dilution of Precision).	Small Integer [2 bytes]	107
GPSHDOP	GPS mode 1, Received from legacy units. Please ignore.	Small Integer[2 bytes]	
GPSVDOP	GPS mode 2, Received from legacy units. Please ignore.	Small Integer[2 bytes]	
GPSHEIGHT	Altitude of current position fix in meters.	Integer	289
NetworkTypeId	The channel of the message (4 - GPRS, 5 - SMS).	Small Integer[2 bytes]	4,5
UnitMsgId	Sequential numerator from the Correlator. Please ignore.	Integer	
InsertDate	Date & Time when an uplink message was inserted to the database.	Date time	
UTCTime	UTC Time from unit's GPS.	Date time	
DriverId	Dallas key (bytes 33-38, see Cellocator protocol).	Long [8 bytes]	3846
TxReasonId	Transmission reason (byte 19, see Cellocator protocol).	Integer	1-255
AlertReason	TX Reason specific data (byte 18, see Cellocator protocol).	Integer	0-255
Hibernation	GSM hibernation bit (No hibernation - 0, in hibernation -1).	Integer	0,1
HRNetwork	Home\Roam network (Home -0 , Roam-1).	Integer	0,1
UnitMode	Used by SVR system. Please ignore.	Integer	0-10
MileCounter	Unit's Odometer value.	Integer	46765

GPSMode1	Defines the validity of GPS data in the message. See Cellocator OTA protocol for further information.	Integer	4
GPSMode2	Defines the validity of GPS data in the message. See Cellocator OTA protocol for further information.	Integer	2
OutputA	Output Gradual Immobilizer (IO byte3, bit 2).	Integer	0,1
OutputB	Output Lights (IO byte4, bit 3).	Integer	0,1
OutputC	Output Immobilizer (IO byte4, bit 5).	Integer	0,1
OutputD	Output Siren (IO byte3, bit 1).	Integer	0,1
OptionalInput	2 analog inputs supported; use the following formulas to extract the values in 1000V: First Analog Input: OptionalInput modulo 2^16. Second Analog Input: OptionalInput / 2^16.	Integer	109774132
GPSCommStatus	Describes status of communication with the GPS module. (0 - not valid, 1 - valid)	Integer	0,1
RawData	The full message according to Cellocator OTA protocol (MCGP or CSA).	VARCHAR[2000]	4D4347500960..
PLMN	Current GSM Operator code. Represents the MCC-MNC of a cellular operator (country code + network number).	Integer	42501
TotalIO	5 bytes of IO, please ignore.	Long [8 bytes]	21474837731
IO_Byte1_bit2	Most common IO bits, please ignore.	Integer	0,1
IO_Byte1_bit3	Most common IO bits, please ignore.	Integer	0,1
IO_Byte1_bit4	Most common IO bits, please ignore.	Integer	0,1
IO_Byte2_bit0	Most common IO bits, please ignore.	Integer	0,1
IO_Byte2_bit1	Most common IO bits, please ignore.	Integer	0,1
IO_Byte2_bit3	Most common IO bits, please ignore.	Integer	0,1
IO_Byte2_bit4	Most common IO bits, please ignore.	Integer	0,1
IO_Byte2_bit5	Most common IO bits, please ignore.	Integer	0,1
IO_Byte2_bit6	Most common IO bits, please ignore.	Integer	0,1
IO_Byte2_bit7	Most common IO bits, please ignore.	Integer	0,1
SN	The Message numerator field contains a value that is increased after every self-initiated generation of a message (in cases where acknowledge from central control was received).	Integer	

MessageType	MCGP: 40 – Forward Message (8); 41 – Modular Message (9), 35 – Programming Ack. (3), 43 – Type 11 [offset of 32 in Cellocator protocol] CSA: according to protocol.	Integer	
UnitData	Uplink data, without Header and Check Sum. Please ignore.	Varchar (500)	
MsgProtocol	OTA protocol type: MCGP = 0 CSA = 1	Small integer [2 bytes]	0,1
TripId	Trip Id in CSA protocol.	Integer	521123
ManeuverId	Maneuver Id in CSA protocol.	Integer	521123
ManeuverUsage	Maneuver data usage in CSA protocol [%].	Small integer [2 bytes]	50%
AccidentBuffer	Accident buffer status bit mask in CSA protocol.	Integer	0-255
ItemId	Item Id is a reference Id between Fleet Log table to Modular Log table for CSA messages.	Long	
CorrelatorAppId	Instance Id of the Correlator corresponds with the CorrelatorAppID in the Correlator INI file.	Integer	0

3.2.2.2 Modular Log Interface

The Modular log data includes extended data that is not fully available in the UplinkMsgLog table.

You can get the data by using the following script:

```
SELECT TOP (1000) [UnitId] ,[ItemId] ,[MsgType] ,[InsertTime] ,[UTCTime] ,[ModuleId]
,[ModuleData] FROM [TWPQueues].[dbo].[ModularLog]
```

Field Name	Field Description	Type	Example
UnitId	Cellocator unit's ID.	Integer	203729
ItemId	Item Id is a reference Id between Fleet Log table to Modular Log table for CSA\Type 11 messages, and for Nano event (Tx. 164) in Type 0.	Long	
Msg Type	CSA: according to CSA wireless protocol. NULL for Type 11.	Integer	0-4
InsertTime	Date & Time when an uplink message was inserted by database.	DateTime	'2009-06-21 11:20:15.720 '
UTCTime	UTC Time from unit's GPS.	Date time	'2009-06-21 11:20:15.720 '
ModuleId	CSA module Id, according to wireless protocol. NULL for Type 11.	Integer	30
ModuleData	All module raw data according to wireless protocol. CSA: will parse the data in XML format. See example in the *Modular data* table. Type 11: parses the data for the following modules - 1, 2, 3, 4, 6, 7, 8, 12, 13, 17, 22, 25, 28, 31, 32, 34, 37, 38, 40, 41, 42, 45, 50, 51, 52, 53, 54. For all other modules, the XML field will include only Module Id and RawData. Nano Event: additional data that appears in the Dallas Code bytes is parsed and inserted into this table. See example in the *Modular data* table.	VARCHAR [8000]	For more information see the following table, *Modular data*.

3.2.2.3 Modular data

The data in the ModuleData parameter of the ModularLog table includes extended information in an XML format; below are examples of the XML format for each type and module of data.

Msg Type	Module ID	Description / Example
CSA	**Module 30** – CSA Full Event	<ModuleData><MMFullEvent><EventReason>1</EventReason> <EventSubReason>46</EventSubReason> <EventNumerator>0</EventNumerator> <OperationMode>0</OperationMode> <StandbyEngineFlag>Off</StandbyEngineFlag> <DrivingFlag>Idling</DrivingFlag> <CalibratingFlag>Ready</CalibratingFlag> <RawLoggingFlag>Off</RawLoggingFlag> <EngineOnFlag>Off</EngineOnFlag> <DriverID>0</DriverID><TripID>0</TripID> <ManeuverID>0</ManeuverID> <ManeuversDataUsage>0</ManeuversDataUsage> <AccidentBufferStatusBitmak>0</AccidentBufferStatusBitmak> <AccidentBufferStatusBitmaskCrash1>Empty </AccidentBufferStatusBitmaskCrash1> <AccidentBufferStatusBitmaskCrash2>Empty </AccidentBufferStatusBitmaskCrash2> <HDOP>0</HDOP><Mode1>0</Mode1> <Mode2>0</Mode2><SatellitesUsed>0</SatellitesUsed> <Longitude>0</Longitude><Latitude>0</Latitude> <Altitude>0</Altitude> <GroundSpeed>0</GroundSpeed> <SpeedDirection>0</SpeedDirection> <Seconds>40</Seconds><Minutes>25</Minutes> <Hours>19</Hours> <Day>12</Day><Month>8</Month> <Year>15</Year></MMFullEvent> <RawData>435341380000000C84E1900001E2E0 12E0000000000000000000000000000000000000000 0000000000000000000000000000 0000000002819130C080F59</RawData> </ModuleData>
11	**Module 41** - Legacy (Type-0) Encapsulated Message + **Module 42**	<PW_Uplink_CAN><Module41><RawData> 00010000000400A6004501010300 03066EEA3C3C7A91002F6B80</RawData> </Module41><Module42><XAcc>-0.022</XAcc> <YAcc>0.150</YAcc><ZAcc>-0.992</ZAcc><Lgt>239.25</Lgt> <Alt>129</Alt><Tmp>27.3</Tmp> <Bat>59</Bat><LRSSI>-52</LRSSI> <RawData>A7FF580280F0BD03AA141101013BCC </RawData></Module42> </PW_Uplink_CAN>
11	**Module 1** - DTC Code	<PW_Uplink_CAN><DTC><I><M>3</M><V>P5</V></I><I><M>7</M><V>c5</V></I></DTC></PW_Uplink_CAN> <I> is for each DTC item, <M> is for the mode 3/7, <V> is the DTC entry value: Type:P/C/B/N and number.
11	**Module 2** - Trigger Event Module	<PW_Uplink_CAN><Trigger_Event><OperatorID>abcd</OperatorID> <PLSignature>1234</PLSignature> <Variables><Variable Value= 1/></Variables></Trigger_Event></PW_Uplink_CAN>

Msg Type	Module ID	Description / Example
11	**Module 3** – OBDII MIL status	<PW_Uplink_CAN><OBD2MIL MilStat="1"/></PW_Uplink_CAN>
11	**Module 4** - Calibration Data Snapshot Module	No XML, just updates MileCounter parameter.
11	**Module 6** - GPS Location Stamp Module	No XML, just updates Message Location parameter.
11	**Module 7** - GPS Time Stamp Module	No XML, just updates Message Date & Time parameter.
11	**Module 8** - Firmware ID Module	No XML, just updates Message HW and SW parameter.
11	**Module 10** - Configuration Memory Write Response	<PW_Uplink_CAN><MemWr Num="Numerator" Cnt="Count"><I V="A"/><I V="0"/>….</MemWr></ PW_Uplink_CAN>
11	**Module 11** - Configuration Memory Read Response	<PW_Uplink_CAN><MemRd Num="Numerator" Cnt="Count"><I Typ="1" Addr="100" lng="2" V="AA"/><I Typ="1" Addr="150" lng="5" V="12345"/>….</MeRdr></PW_Uplink_CAN>
11	**Module 12**, Name: CAN-GPS Calibration Status	<PW_Uplink_CAN><Module12><CalbState>0</CalbState><ConvMatrix> 6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35,36,37,38,3 </ConvMatrix><RawData>…..</RawData></Module12></PW_Uplink_CAN>
11	**Module 22** - VIN String Write Module	<PW_Uplink_CAN><VIN>abcd…</VIN></PW_Uplink_CAN>
11	**Module 25** - Trigger Event ID Module	<PW_Uplink_CAN><Trigger_Event_Op_ID>abcd</Trigger_Event_Op_ID> </PW_Uplink_CAN>
	Module 28 - General Status Event	<PW_Uplink_CAN><Module28><RSSI>-80</RSSI><MAC>481A84001C76</MAC><OTAVer>1</OTAVer><Bat>2.974</Bat><BOMMask>DD5C</BOMMask><TR>2</TR><FWVer>04.63</FWVer><Tmp>25.3</Tmp><Lgt>450.5</Lgt><Mgnt>False</Mgnt><XAcc>0</XAcc><YAcc>-0.064</YAcc><ZAcc>-0.96</ZAcc><AccSTst>Pass</AccSTst><Pkg>Close</Pkg><THState>7</THState><RawAdv>B0481A84001C76019E0B5CDD02630403</RawAdv><RawSnsRead>FD00FFFF850382000000FF00F19E0B07B0</RawSnsRead><RawData>0101000D0022FCB0481A84001C76019E0B5CDD02630403FD00FFFF850382000000FF00F19E0B07B0</RawData></Module28></PW_Uplink_CAN>
11	**Module 31** - CAN Variables Status Dump	<Trigger_Event_Vars><Variable VarID= x Value= y/> <Variable VarID= z Value= w/>….</Trigger_Event_Vars>
11	**Module 34** - TPA Event	<PW_Uplink_CAN><Module34><TripID>xx</TripID><ZoneID>zz</ZoneID><ZoneFee>f.f</ZoneFee><TpaTR>xx</TpaTR><Diag>xx</Diag><OdomKM>f.f</OdomKM><TripDistKM>f.f</TripDistKM><TripFee>f.f</TripFee><ZoneDistKM>f.f</ZoneDistKM><LastFee>f.f</LastFee><Changes>xx</Changes><RawData></RawData></Module34></PW_Uplink_CAN>

Msg Type	Module ID	Description / Example
11	**Module 37** - Current J1939 DTC Status	<PW_Uplink_CAN><Module37><Source No="x"><IndLamp PL="x" AW="x RS="x" MI="x"><DTC><I SPN="x" FMI="x" OC="x" CM=X/><I SPN="x" FMI="x" OC="x" CM=X/>...</DTC></Source><RawData></RawData></Module37></PW_Uplink_CAN>
11	**Module 38** - J1939 DTC Appeared / Disappeared	<PW_Uplink_CAN><Module38><TR_BM>bb</TR_BM><SrcNo>X</SrcNo><IndLamp PL="x" AW="x RS="x" MI="x"><DTC><I SPN="x" FMI="x" OC="x" CM=X/><I SPN="x" FMI="x" OC="x" CM=X/>...</DTC><RawData></RawData></Module38></PW_Uplink_CAN>
11	**Module 40** - Measurement readings	<PW_Uplink_CAN><Module40><SrcEvnt>253</SrcEvnt><NanoTxR>0</NanoTxR><Status>0</Status><Start/><Chrg>0</Chrg><UpTH>30</UpTH><LoTH>255</LoTH><Tmp><I><V>28.1</V><T>02/07/2018 16:17:48</T></I><I><V>28.1</V><T>02/07/2018 16:18:48</T></I><I><V>27.9</V><T>02/07/2018 16:19:48</T></I><I><V>28</V><T>02/07/2018 16:20:48</T></I><I><V>28.1</V><T>02/07/2018 16:21:48</T></I><I><V>28.1</V><T>02/07/2018 16:22:48</T></I><I><V>27.9</V><T>02/07/2018 16:23:48</T></I><I><V>29.1</V><T>02/07/2018 17:30:48</T></I><I><V>28.9</V><T>02/07/2018 17:31:48</T></I><I><V>28.7</V><T>02/07/2018 17:32:48</T></I><I><V>28.9</V><T>02/07/2018 17:33:48</T></I><I><V>28.9</V><T>02/07/2018 17:34:48</T></I><I><V>28.7</V><T>02/07/2018 17:35:48</T></I><I><V>28.7</V><T>02/07/2018 17:36:48</T></I><I><V>28.7</V><T>02/07/2018 17:37:48</T></I><I><V>28.9</V><T>02/07/2018 17:38:48</T></I><I><V>28.6</V><T>02/07/2018 17:39:48</T></I><I><V>28.7</V><T>02/07/2018 17:40:48</T></I><I><V>28.7</V><T>02/07/2018 17:41:48</T></I><I><V>28.6</V><T>02/07/2018 17:42:48</T></I><I><V>28.7</V><T>02/07/2018 17:43:48</T></I><I><V>28.6</V><T>02/07/2018 17:44:48</T></I><I><V>28.7</V><T>02/07/2018 17:45:48</T></I><I><V>28.6</V><T>02/07/2018 17:46:48</T></I><I><V>28.6</V><T>02/07/2018 17:47:48</T></I><I><V>28.6</V><T>02/07/2018 17:48:48</T></I><I><V>28.6</V><T>02/07/2018 17:49:48</T></I><I><V>28.4</V><T>02/07/2018 17:50:48</T></I><I><V>28.5</V><T>02/07/2018 17:51:48</T></I><I><V>28.6</V><T>02/07/2018 17:52:48</T></I></Tmp><RawData>010001000D0000FD3011100207123C001EFF031E1901011901011701011801011901011901011701012301432101011F0101210101210101F01011F01011F01011F01012101011E01011F01011F01011E01011F01011E01011F01011E01011E01011E01011E01011C01011D01011E0101</RawData></Module40></PW_Uplink_CAN>
11	**Module 42** - Nano Inherent Sensors	<PW_Uplink_CAN><Module42><XAcc>-0.074</XAcc><YAcc>0.048</YAcc><ZAcc>-0.976</ZAcc><Lgt>0.25</Lgt><Alt>128.4</Alt><Tmp>28.6</Tmp><Bat>78</Bat><LRSSI>-52</LRSSI><RawData>D7FEC000C2F00100A4141E01A04ECC</RawData></Module42></PW_Uplink_CAN>

Msg Type	Module ID	Description / Example
11	**Module 44** - MultiSense Additional Information	<PW_Uplink_CAN><Module44><MAC>481A84001950</MAC><Bat>25</Bat><RSSI>-49</RSSI><LastComm>2018/07/02 21:08:33</LastComm><MSFW>04.61</MSFW><BOMMask>7DFD</BOMMask><Tmp>26.8</Tmp><Hum>54.6</Hum><Lgt>0</Lgt><XAcc>-0.192</XAcc><YAcc>-0.960</YAcc><ZAcc>-0.128</ZAcc><Mgnt>False</Mgnt><Pkg>Close</Pkg><AccSTst>Pass</AccSTst><RawData>481A8400195019CF21081502071261O4FD7D0C012202000000FD00F100FE80</RawData></Module44></PW_Uplink_CAN>
11	**Module 45** - Full System MultiSense Readings	<PW_Uplink_CAN><Module45><TxR>0</TxR><Idx>0</Idx> <MAC>481A84000AF9</MAC><MsTx>4</MsTx><ThSt>0</ThSt> <Bat>100</Bat><RSSI>-40</RSSI><Prtc>1</Prtc> <MSFW>04.5B</MSFW> <BOMMask>5CFD</BOMMask><Tmp>27</Tmp> <Hum>-0.1</Hum><Lgt>199</Lgt><XAcc>0.000</XAcc> <YAcc>0.064</YAcc><ZAcc>-0.960</ZAcc><Mgnt>False</Mgnt> <Pkg>Open</Pkg> <AccSTst>Pass</AccSTst> <RawData>010000481A84000AF9040064D8015B 04FD5C0E01FFFF8E010000000100F18200000000 </RawData></Module45></PW_Uplink_CAN>
11	**Module 50** - DTCO Connect / Disconnect Event	<PW_Uplink_CAN><Module50><OdoSrc>3</OdoSrc><DrvSrc>0</DrvSrc> <RpmSrc>2</RpmSrc> <SpdSrc>3</SpdSrc><D8CS>0</D8CS> <SWNo>0</SWNo><CP>1</CP> <VRN>ABC123 </VRN><WS>75</WS> <RawData>033800000141424331323320202020202020204B</RawData> Module50></PW_Uplink_CAN>
11	**Module 51** - DTCO Time	<PW_Uplink_CAN><Module51><Time>2031/06/15 11:09:28</Time> <RawData>001C090B0F061F</RawData> </Module51></PW_Uplink_CAN>
11	**Module 52** - DTCO Driver Identification Numbers	<PW_Uplink_CAN><Module52><Din1>000123071400321300033800000141424331</Din1> <Din2>3233202020202020204B0613000504020D2D</Din2> <RawData>01123131303030303030303833333230303001123131303030303030303833333830303O</RawData></Module52></PW_Uplink_CAN>
11	**Module 53** - DTCO Parameter Change Event	<PW_Uplink_CAN><Module53><OdoSrc>3</OdoSrc><DrvSrc>0</DrvSrc> <RpmSrc>2</RpmSrc><SpdSrc>3</SpdSrc><D8CS>0</D8CS> <D8VDO>1</D8VDO><D8SR>0</D8SR><FMS>0</FMS> <GRD1>0</GRD1><GRD2>0</GRD2><GRWS>0</GRWS> <GRTc>0</GRTc><GRSp>0</GRSp><GRAI>0</GRAI><GRVIN>0</GRVIN><GRDIN1>0</GRDIN1><GRDIN2>0</GRDIN2><GRDC>0</GRDC><GRDD>1</GRDD> <GRRq>0</GRRq> <WS>75</WS><D1S>16</D1S><D2S>208</D2S><TcS>193</TcS><SpdAu>0</SpdAu> <AddIn>20481</AddIn><RawData>0338020000404B10D0C100000150</RawData></Module53></PW_Uplink_CAN>
11	**Module 54** - DTCO Periodic Event	<PW_Uplink_CAN><Module54><OdoSrc>x</OdoSrc><DrvSrc>x</DrvSrc><RpmSrc>x</RpmSrc><SpdSrc>x</SpdSrc><D8CS>x</D8CS><Spd>x</Spd><OdoKM>f.f</OdoKM><TrpKM>f.f"</TrpKM><KFac>f.f</KFac><EngSpd>f.f</EngSpd><RawData></RawData></Module54></PW_Uplink_CAN>
11	**Module 60** - TDLT Event	<PW_Uplink_CAN><Module60><LgIn>x</LgIn><CrdVld>x</CrdVld><VclTyp>x</VclTyp><RSSI>x</RSSI><XAcc>f.f</XAcc><YAcc>f.f</YAcc><ZAcc>f.f</ZAcc><Crd1>s</Crd1><Crd2>s</Crd2>"<Crd3>s"</Crd3><RawData></RawData></Module60></PW_Uplink_CAN>

Msg Type	Module ID	Description / Example
11	Unrecognize module	<PW_Uplink_CAN><Modulexxx>… <BadRawData></BadRawData></Modulexxx></PW_Uplink_CAN>
0	Nano Event: On bytes 33-38 when extended datatype is 5 or TXReason is 164.	<Nano><XAcc>0</XAcc> <YAcc>0</YAcc><ZAcc>-0.75</ZAcc><Ori>A</Ori> <Lgt>248.5</Lgt><Alt>112</Alt> </Nano>.

3.3 Advanced Integration: Message Parsing

This section describes the message parsing for both MCGP and CSA messages.

Note the following:

Byte number of message is 1 based (range 1 to length of message).

Bit number in byte is 0 based (range 0-7).

3.3.1 *Common Parameters for MCGP and CSA Messages*

Common Parameters for MCGP Messages

Description	CellocatorHub
Message data from MCGP header to the checksum – including both.	RawData
Message type (byte 5)	MessageType
Unit ID (Bytes 6-9)	CMUId
Communication Control (Byte 10 bit 0)	MessageInitiative 0 – Active transmissions (initiated by the unit, based on its logic and decisions) 1 – Passive responses (response to a command or a query message)
Communication Control (Byte 10 bit 3)	MessageSource 0 – Direct message (not from memory) – Alert Note that the only exception is the "Transmission Reason 32 - IP changed/Connection up" message, which always requires ACK from the server, even if it was sent as a direct message and not through memory. 1 – Message from memory (the unit tries to resend the message from the memory, until ACK from the server is received)
Message numerator (byte 12)	Numerator
Source of message according to message metadata.	NetworkTypeId 5 – IIS (SMS) 4 – GPRS
Protocol identifier	1 for MCGP, 256 for CSA (Safety)

Common Parameters for CSA Messages

Description	CellocatorHub
Message data from CSA header to the checksum – including both.	RawData
Packet Control (Byte 8-bits 0-2)	MessageType

	0 – Outbound message 1 – Ack 2 – Programming Command 3 – Programming Response 4 - Request
Unit ID (Bytes 9-12)	CMUId
Packet Control (Byte 8 bit 7)	MessageInitiative 0 – Unit initiated 1 – Reply or Ack Note: CorrelatorMax inverts the original bit in order to keep compliance with MCGP protocol.
Communication Control (Byte 10 bit 3)	MessageSource 0 – Direct message (not from memory) – Alert Note that the only exception is the "Transmission Reason 32 - IP changed/Connection up" message, which always requires ACK from the server, even if it was sent as a direct message and not through memory. 1 – Message from memory (the unit tries to resend the message from the memory, until ACK from the server is received)
Message numerator (byte 6-7)	Numerator
Protocol identifier	Protocol = 256 for CSA (Safety)

3.3.2 *MCGP Messages*

3.3.2.1 Status/Location Message (Message Type 0)

Byte	Description	TWPQueues	CellocatorHub
1	MCGP		Protocol = 1 (1= MCGP, 256 = CSA)
2			
3			
4			
5	Message Type (0)		MessageType = 0
6	Unit ID		CMUId
7			
8			
9			
10	Communication Control Field		MultiPurposeType Bits 0-1 (Bits 5-6)
11			Hibernation (Bit 7) FW Ver – Minor version (Bits 0-4)
12	Message Numerator (Anti-Tango™)		Numerator
13	Unit Hardware Version		HWID
14	Unit Firmware Version		FWVer - Main version
15	Protocol Version and Unit Functionalities		
16	Unit Status and Current GSM Operator (1st Nibble)		GPSCommStatus (Bit 0 inverted) HRNetwork (Bit 1) – 0: home network, 1 - roaming PLMN Bits 12-16 (Bits 4-7)
17	Current GSM Operator (2nd and 3rd Nibbles)		PLMN Bits 8-15
18	Transmission Reason Specific Data		TRSpecificData
19	Transmission Reason	See the following TWPQueues Alert reasons logic section.	TRId
20	Unit Mode of Operation		UnitMode
21	Unit I/O Status 1st byte		TotalIO See the following I/O table mapping section.
22	Unit I/O Status 2nd byte		
23	Unit I/O Status 3rd byte		
24	Unit I/O Status 4th byte		

Byte	Description	TWPQueues	CellocatorHub	
25	Current GSM Operator (4th and 5th Nibbles)		PLMN Bits 0-7	
26	Analog Input 1 Value		Analog1	Values are floating point, converted according to HW type and settings.
27	Analog Input 2 Value		Analog2	
28	Analog Input 3 Value		Analog3	
29	Analog Input 4 Value		Analog4	
30	Mileage Counter (Odometer)		Odometer	
31				
32				
33	Multi-Purpose Field (Driver/Passenger/Group ID, PSP/Keyboard Specific Data, Accelerometer Status, SIM IMSI)		MultiPurpose See the following MultiPurpose table section.	
34				
35				
36				
37				
38				
39	Last GPS Fix		LastGPSFixDay Note: If time is in future, reduce by one month.	
40				
41	Service and Status		MultiPurposeType Bit 2 (Bit 7)	
42	Mode 1		GPSMode1	
43	Mode 2		GPSMode2	
44	Number of Satellites Used		NumOfSat	
45	Longitude		Long (Decimal Degrees)	
46				
47				
48				
49	Latitude		Lat (Decimal Degrees)	
50				
51				
52				
53	Altitude		Alt (meters)	
54				

Byte	Description	TWPQueues	CellocatorHub
55			
56			
57	Ground Speed		GPSSpeed (KM/H)
58			
59			
60			
61	Speed Direction (True Course)		GPSCourse (Degrees)
62			
63	UTC Time – Seconds		UTCTime GPSDateTime Note: If time is in future or "bad", set to: 1:1:1 1/1/2007
64	UTC Time – Minutes		
65	UTC Time – Hours		
66	UTC Date – Day		
67	UTC Date – Month		
68	UTC Date – Year (-2000) (e.g. value of 7 = year 2007)		
69			
70	Error Detection Code (8-bit additive checksum, excluding system code)		

Multi-Purpose Field

According to the `MultiPurposeType` and `TRId` the parameters and values change, as listed in the table below:

TRId	Byte 41	Byte 10		Data in Bytes 33-38
	Bit 7	Bit 5	Bit 4	
202	X	X	X	Sim
12	X	X	X	1-Wire `MultiPurpose = <OneWire SensorId="TRSubReason LowNibble" State="TRSubReason High Nibble">Byte33-36</OneWire>`
	0	0	0	`DriverID` (Byte33-Byte38)

TRId	Byte 41	Byte 10		Data in Bytes 33-38
	Bit 7	Bit 5	Bit 4	
	0	0	1	If Byte 33 = 1, i.e Bad External Alarm device data: MultiPurpose = <PSP><CommStatus>Bad</CommStatus></PSP> Otherwise: MultiPurpose = <PSP><CommStatus>OK</CommStatus><AlarmStatus>*Byte33Bit4 to Byte35Bit7*</AlarmStatus></PSP>
	0	1	0	If Byte 33 = 1, i.e Bad External Alarm device data: MultiPurpose = <Keyboard> <CommStatus>Bad</CommStatus> </Keyboard> Otherwise: MultiPurpose = <Keyboard> <CommStatus>OK</CommStatus> <AlarmStatus>*Byte33Bit4toByte35Bit7*</AlarmStatus> <Door>*Byte33Bit4*</Door> <Volume> *Byte33Bit5-6*</Volume> <Ignition>*Byte33Bit7*</Ignition> <AlarmArmed> *Byte34Bits1-2*</AlarmArmed> <ImmoArmed>*Byte34Bits4-5*</ImmoImmo> <HotWiring>*Byte34Bit6*</HotWiring> <Service> *Byte34Bit7*</Service> <WrongCode>*Byte35Bit1*</WrongCode> </Keyboard>
	0	1	1	MultiPurpose = <TrailerID>*Byte33-Byte38*</TrailerID>
	1	0	0	MultiPurpose = <IMEI>*Byte33-Byte38 + Byte41Bits5-6 as MSBits*</IMEI>
	1	0	1	MultiPurpose = see the following *CelloTrack Nano Data* section.
Otherwise				MultiPurpose = "" (empty string)

CelloTrack Nano Data MultiPurpose data table

The following XML parameters are added to the `MultiPurpose` parameter under `<MltSns>` or `<Nano>` tag. Each parameter is added according to the condition of the `TRSubReason` and `ManagementByte` values.

TRSubReason	ManagementByte (Byte 33)	XML	Remarks
	0	<XAcc>*Byte34Bits4-7*</XAcc> <XAcc>*Byte34Bits0-3*</XAcc> <XAcc>*Byte35Bits4-7*</XAcc> <Ori>*Byte35Bit7*</Ori> <InvalidTmp/> *if Byte35Bit1=0* <PositionUpright/> *if Byte35Bit2-3=0* <PositionLaying/> *if Byte35Bit2-3=2*	Acceleration in G (multiplier 0.25), Ori is A or B. <InvalidTmp/>,<PositionUpright/>/<PositionLaying/> - are optional tags that are added according to byte 15 flags. <PositionUpright/>/<PositionLaying/> are the same as <Ori>A/B</Ori>
	1-256	<SrcEvnt>Byte34</SrcEvnt>	
11,12	1	<Status>*Byte35Bits0-2*</Status> <End/> or <Start/> according to Byte35 Bit 7 <Chrg> *Byte35Bits4-5* </Chrg>	
4,5,10,11,12, 14,15	2	<PrbCode>*Byte36*</PrbCode>	
1,4,5,10,11,12,16		<Batt>*Byte36*</Batt> <LRSSI>*Byte37*</LRSSI>	
14,15	2	<Batt>*Byte36*</Batt> <LRSSI>*Byte37*</LRSSI>	
Not 1,4,5,10,11,12,14 ,15		<Lgt>*Byte36+Byte37Bits0-3*</Lgt>	Lux (multiplier 0.25)
1,16		<AccRMS>*Byte38*</AccRMS>	RMS (multiplier 0.032)
4,5		<Lgt>*Byte38*</Lgt>	In Lux – Nano multiplier 2, Nano multiplier 4
11,12	2	<Status>*Byte38Bits0-2*</Status> <End/> or <Start/> according to Byte35 Bit 7	
14,15	Not 2	<Alt>*Byte38*</Alt>	In meters (-400 to 7760)

TWPQueues logic

Some logic is applied when using TWPqueue; the following @alertReason are created according to the logic:

Value	Condition
DriverID=-1	Cellotrack HW (123,172,23,12)
AnalogInput1 & AnalogInput2 are signed values (-128 to 127)*Coeff	For HWs (26,58,90,122,154,186,218,250,8,40,72,136,168,200)
@GPSGeoX, @GPSGeoY converted	LoadConversionDll=true in ini
GPSMode1 = 99, GPSMode2 = 99	GPSMode99Enabled && (!bTimeInaccurate) && (GPSmode1 == 0) && (GPSmode2 == 0) && ((Latitude != 0) || (Longitude != 0))
@IsAlert=0 (false)	TxReason 11,44
@EngineOn=1 (true)	TxReason 69
@EngineOn=0 (false)	TxReason 53
@ExtOutputB = LedOut	For CR00 HWs (221,222,29,30)
@ExtOutputB = UnlockIn	For no CR200 HWs
Invert @GPSPower	GpsPowerCheck enable in ini and HW is CelloTrack (23,12,172,183)
@GPSSpeed=0	!((GPSMode1 == 3 || GPSMode1 == 4) && GPSMode2 == 2) || this.GPSpower == 1

3.3.2.2 Logged Fragment of Forwarded Data from Serial Port to Wireless Channel (Message Type 7)

Byte	Description	TWPQueues	CellocatorHub
1	MCGP		Protocol = 1 (1= MCGP, 256 = CSA)
2			
3			
4			
5	Message Type (7)		MessageType = 7
6	Unit ID		CMUId
7			
8			
9			
10	Communication Control Field		
11			Hibernation (Bit 7) FW Ver – Minor version (Bits 0-4)
12	Message Numerator (Anti-Tango™)		Numerator
13	Serial Port Source		
14	Forwarded Message Code Sequential 7 bits ID of the container + container indication bit (MSB) Assigned for each container		
15	Fragment Control Byte		
16-69	Container Fragment (First fragment begins with two bytes of length of container, last one is zero padded)		modularData: `<Forward><MDT Type="0" Id="" FwdType="" Result="0" Time=""/><Payload>PayloadData</Payload></Forward>` See the following *Container Fragment* section.
70	Error Detection Code (8-bit additive checksum, excluding system code)		

3.3.2.3 Container Fragment

The container is a data structure, created by the unit in its RAM buffer upon reception of the data for forwarding from the unit serial port (if enabled in the "Forward Data as Container" parameter (address 285, bit 6)).

The forwarded payload from serial port is escorted by 48 bytes of FM (Fleet Management) data, and 2 bytes of total length of payload + FM data.

Every container is assigned by a 7-bit numerator (increased every data packet received from the serial port), used in fragmentation process and reported with the container.

The container data structure is described below:

- RawData parameter includes raw data of multiple messages that formed the container.
- Payload is added into the ModularData parameter:
  - `<MDT>` tag includes data extracted from payload and configuration with the following attributes:
    - `Type` - `NAVMAN = 1, GARMIN = 3, MICRONET = 2, TBOX = 4`
    - `ID` – extracted from `GARMIN` / `NAVMAN` message
    - `FwdType` – extracted from `GARMIN` / `NAVMAN` message
    - `Result` – extracted from `GARMIN` / `NAVMAN` message
    - `Time` – extracted from `GARMIN` message
  - `<Payload>` - Payload data as hexadecimal string, after conversion if needed (remove delimiters, control characters, etc).

Byte	Description	TWPQueues	CellocatorHub
1-2	Payload length (X)		
3 to 3+x	Forwarded Payload from serial port, X bytes (up to 512 bytes)		`<Forward><MDT Type="0" Id="" FwdType="" Result="0" Time=""/><Payload>PayloadData</Payload></Forward>` See description above.
4+X	Unit Status + Current GSM Operator (1st nibble) (same as byte 16 of type 0)		GPSCommStatus (Bit 0 inverted) HRNetwork (Bit 1) PLMN Bits 12-16 (Bits 4-7)
5+X	Current GSM Operator (2nd and 3rd nibbles) (same as byte 17 of type 0)		PLMN Bits 8-15
6+X	Current GSM Operator (4th and 5th nibbles) (same as byte 25 of type 0)		PLMN Bits 0-7
7+X	Unit Mode of Operation (same as byte 20 of type 0)		UnitMode
8+X	Unit I/O Status 1st byte (same as byte 21 of type 0)		`TotalIO`

Byte	Description	TWPQueues	CellocatorHub	
9+X	Unit I/O Status 2nd byte (same as byte 22 of type 0)		See the *I/O table mapping* section.	
10+X	Unit I/O Status 3rd byte (same as byte 23 of type 0)			
11+X	Unit I/O Status 4th byte (same as byte 24 of type 0)			
12+X	Analog Input 1 value (same as byte 26 of type 0)		Analog1	(Values are floating points, converted according to HW type and settings)
13+X	Analog Input 2 Value (same as byte 27 of type 0)		Analog2	
14+X	Analog Input 3 Value (same as byte 28 of type 0)		Analog3	
15+X	Analog Input 4 Value (same as byte 29 of type 0)		Analog4	
16+X	Mileage Counter (Odometer) (same as bytes 30-32 of type 0)		Odometer	
17+X				
18+X				
19+X	Multi-Purpose Field (Driver/Passenger/Group ID, PSP/Keyboard Specific Data, Accelerometer Status, SIM IMSI) (same as bytes 33-38 of type 0)		MultiPurpose See the *MultiPurpose Field* section.	
20+X				
21+X				
22+X				
23+X				
24+X				
25+X	Last GPS Fix (same as bytes 39-40 of type 0)		`LastGPSFixDay` Note: If time is in future, reduce by one month.	
26+X				
27+X	Location Status (flags) (same as sub type 4 of type 9)		MultiPurposeType Bit 2 (Bit 7) Note: MultiPurposeType Bits 0-1 (Bits 5-6) is taken from communication control byte in message header.	
28+X	Mode 1		GPSMode1	
29+X	Mode 2		GPSMode2	
30+X	Number of Satellites Used		NumOfSat	
31+X	Longitude		Long (Decimal Degrees)	

Byte	Description	TWPQueues	CellocatorHub
32+X			
33+X			
34+X			
35+X	Latitude		Lat (Decimal Degrees)
36+X			
37+X			
38+X			
39+X	Altitude		Alt (meters)
40+X			
41+X			
42+X	Ground speed		GPSSpeed (KM/H)
43+X			
44+X	Speed direction (true course)		GPSCourse (Degrees)
45+X			
46+X	UTC time – Seconds		UTCTime GPSDateTime Note: If time is in future or "bad", set to 1:1:1 1/1/2007
47+X	UTC time – Minutes		
48+X	UTC time – Hours		
49+X	UTC date – Day		
50+X	UTC date - Month		
51+X	UTC date - Year (-2000) (e.g. value of 7 = year 2007)		

3.3.2.4 Real Time Forwarded Data from Serial Port to Wireless Channel (Message Type 8)

Byte	Description	TWPQueues	CellocatorHub
1	MCGP		Protocol = 1 (1= MCGP, 256 = CSA)
2			
3			
4			
5	Message Type (8)		MessageType = 8
6	Unit ID		CMUId
7			
8			
9			
10	Message Numerator (Anti-Tango™)		Numerator
11	Spare		
12	Spare		
13	Serial Port Source		
14	Spare		
15	Forwarded Message Code		
16	Fragment Control Byte		
17	Payload length		
18			
19 …	Payload		ModularData= <Payload>all payload bytes</Payload> See the *Container Fragment* section.
..	Error Detection Code (8-bit additive checksum, excluding system code)		

3.3.2.5 Modular Message (Message Type 9)

Byte	Description	TWPQueues	CellocatorHub
1	MCGP		Protocol = 1 (1= MCGP, 256 = CSA)
2			
3			
4			
5	Message Type (9)		MessageType = 9
6	Unit ID		CMUId
7			
8			
9			
10	Communication Control Field		MultiPurposeType Bits 0-1 (Bits 5-6)
11			Hibernation (Bit 7) FW Ver – Minor version (Bits 0-4)
12	Message Numerator (Anti-Tango™)		Numerator
13	Packet Control Field		
14	Length (of the modules section - not including the checksum)		
15	First Sub-data Type		ModularData - see the following *Modular data table* section. Modules – a ';' separated list of module type IDs.
16	First Sub-data Length		
17	First Sub-data The Data		
..	…		
..	Error Detection Code (8-bit additive checksum, excluding system code)		

3.3.2.6 Modular data

Note that in TWPQueues this Modular Data for `<Type9>` messages are a separate table; in CellocatorHub it is in the "ModularData" parameter/column.

- Each module generates is parsed to relevant parameters in CellocatorHubDB.
- Additional data is transferred as XML; in the "ModularData" column, the "Modules" column includes a ";" separated list of modules IDs detected.
- XML's of all modules are surrounded by a `<Type9>` tag for CellocatorHub, and with `<PW_Uplink_CAN>` tag for TWPQueues.
- Byte number in the following table is from the start of the module: Byte0 is the module ID, Byte1 is the module length, and Byte2 is the first byte of module data.
- For TWPQueues, AlertReason parameter will be set to 900 + Module id (for example, module 13 AlertReason = 913).

Id	Name	XML tag + CellocatorHub parameters	Remarks
1	Firmware Platform Manifest	`<FwMnf PrFm=Byte2 CPU=Byte3 PrMem=Byte4-5 VMem=Byte6-7 IntNVM=Byte8-9 ExtNVM=Byte10-11 ExtTyp=Byte12 HW=Byte13 Rprg=Byte14-15 SLV=Byte16-17 FW=Byte18-19/>`	
2	CAN data (CompactCAN)	`<Data><Type>3</Type><TrigNo>X</TrigNo> <SensNo>Y</SensNo><SensVal>Z</SensVal> </Data>`	X is the trigger number Y is the sensor number Z is the extraction of the value
3	CAN trigger type + complex trigger (CompactCAN)		
4	Time & location stamp	GPSCommStatus=Byte2Bit6 GPSMode1=Byte3 GPSMode2=Byte4 NumOfSat=Byte5 Long=Byte6-9 Lat=Byte10-13 Alt=Byte14-16 GPSSpeed=Byte17-18 GPSCourse=Byte19-20 UTCTime=Byte21-26 GPSDateTime=Byte21-26 `<GPS><X>PosX</X><Y>PosY</Y><Mode1>Mode1</Mode1><Mode2>Mode2</Mode2><Speed>Speed</Speed><Dir>Direction</Dir><TimeUTCTime</Time></GPS>`	UTCTime - If time is in time or "bad", UTCTime is set to: 1:1:1 1/1/2007 GPSDateTime is set to null if timeInaccuarcy flag at Byte2Bit7 is 1. TWPQueues @GPSCommStatus=Byte2Bit6 PosX, PosY – Calculated X&Y position in Rad * 10-8, values may pass convertion (if LoadConversionDll is set) Mode1&Mode2 –GPS mode, if time is inaccurate and mode1&2 are 0 and GPSMode99Enabled is set, the value will be 99.
6	T Command		@UnitData = Module payload.
7	Usage counters		@UnitData = Module payload.

Id	Name	XML tag + CellocatorHub parameters	Remarks
10	Maintenance message		@UnitData = Module payload.
11	Message Forwarded from Keyboard		@UnitData = Byte1 + Byte4-27
13	Compressed vector change (Curve smoothing)	TRId=913 TXIndex=0-… Long=Byte3-6 Lat=Byte7-10 Odometer=Byte11-13 GPSCourse=Byte15 Speed=Byte16 UTCTime=Byte21-26 + Current year and month.	A set of rows will be added, one for each vector including the first one. RawData of all rows will be the same. TxIndex will indicate the vector number (base is 0, 1 is first change…_). Odometer for first vector is correct, other vectors odometers are set to null (or -1 in TWPQueues) For TWPQueues only: @EngineOn=1 @GPSCommExist=1 @GPSMode1=4 @GPSMod2=2
18	Modular Platform Manifest	<MdlMnf Fx=Vx Fy=Vy…. />	Fx stand for Field number x, Vx stands for the value as hexadecimal string. For more details on the fields and values see "Modular Platform Manifest" in the "Wireless Protocol" document.
20	Pulse Counter Measurement Response	<Module20><Liter_Counter_1>*Byte4-7*</Liter_Counter_1> <Liter_Counter_2> *Byte8-11*<Liter_Counter_2><Module20> "</Module" + type.ToString() + ">";	Value of counter as unsigned int32
22	Cello-CANiQ Fleet End of Trip Report	<EndTrp Odm=*Byte4-7* Cons=*Byte8-11* LvlLt=*Byte12-13* LvlPr=*Byte14* EngMin=*Byte15-16*/>	Odm=Odometer Cons=Trip Fuel Consumption LvlLt= Fuel Tank Level (Liters) LvlPr= Fuel Tank Level (%) EngMin= Delta engine hours in this trip (Minutes)
24	CFE inputs update message	<CFE_IO> <Item Idx=*InputIndex* Typ InputType Fnc=InputFunction Value=*Value adjusted according to input time* <Item … </CFE_IO>	InputType: 0 - Discrete Dry 1 - Discrete Wet 2 - 8 bits Analog measurement (0-2.5 v) 3 - 8 bits Analog measurement (0-30 v)

Id	Name	XML tag + CellocatorHub parameters	Remarks
			4 - Frequency meter 5 - Pulse Counter 6 - 12 bits Analog measurement (0-2.5 v) 7 - 12 bits Analog measurement (0-30 v)
25	oneWire Temperature Sensor	<OneWire> <OW1 + " ID=Byte0-3 V=Byte6-7/> <OW2 + " ID=Byte8-11 V=Byte12-13/> <OW3 + " ID=Byte14-17 V=Byte18-19/> <OW4 + " ID=Byte20-23 V=Byte24-25/> </OneWire>	OWx – Index of 1 wire 1-4 V - is an adjusted One-Wire measurement (LSB) (Coefficient 0.0625)
9	CellId message (2G)	<Cell> <CellType>GSM</CellType> <Time>*Byte3-5*</Time> <Date>*Byte6-8*</ Date > <BSIC>*Byte9*</BSIC> <LAC>*Byte10-11*</LAC> <CellID>*Byte12-13*</CellID> <RSSI>*Byte14*</RSSI> <Neighbors> <N*x*> <BSIC>*Byte(15+6x)*</BSIC> <LAC>*Byte(16+6x)-Byte(17+6x)*</LAC> <CellID>*Byte(18+6x)-Byte(19+6x)*</CellID> <RSSI>*Byte(20+6x)*</RSSI> </N*x*> <Ny … </Neighbors> </Cell> UTCTime=Byte3-8	
12	3G Cell ID Data	<Cell> <CellType>3G</CellType> <Time>*Byte3-5*</Time> <Date>*Byte6-8*</Date> <MCC>*Byte9-10*</MCC> <MNC>*Byte11-12*</MNC> <LAC>*Byte13-14*</LAC> <RSCP>*Byte15*</RSCP> <CellID>*Byte16-19*</CellID>	AcT (Access Technology): 0 – GSM 2 – UTRAN MCC (Mobile Country Code, Decimal, 200-901) MNC (Mobile Network Code, Decimal, 0-999) LAC (Localization Area Code) RSCP (Received Signal Code Power)

Id	Name	XML tag + CellocatorHub parameters	Remarks
		<RSSI>*Byte20*</RSSI> <AcT>*Byte21*</AcT> <PSC>*Byte22-23*</PSC> </Cell> UTCTime=Byte3-8	PSC (Primary Scrambling Code)
28	CDMA Cell ID Data	<Cell> <CellType>CDMA</CellType> <Time>*Byte3-5*</Time> <Date>*Byte6-8*</Date> <SID>*Byte9-10*</SID> <NID>*Byte11-12*</NID> <RSSI>*Byte13*</RSSI> <BSID>*Byte14*</BSID> </Cell> UTCTime=Byte3-8	SID (System ID) NID (Network ID) BSID (Base Station ID (Cell ID + possible sector))
29	4G Cell ID Data	<Cell> <CellType>4G</CellType> <Time>*Byte3-5*</Time> <Date>*Byte6-8*</ Date > <TAC>*Byte9*</TAC> <GCellID>*Byte11-13*</GCellID> <CellID>*Byte14-15*</CellID> <SRXLEV>*Byte16*</SRXLEV> <RSRP>*Byte17*</RSRP> <Neighbors> <N*x*> <CellID>*Byte(20+6x)-(21+6x)*</CellID> <RSRP>*Byte(22+6x)*</RSRP> <SRXLEV>*Byte(23+6x)*</SRXLEV> <RSSI>*Byte(24+6x)*</RSSI> </N*x*> <Ny … </Neighbors> </Cell> UTCTime=Byte3-8	GCellID - Serving Cell Global Cell ID CellID - Serving Cell Physical Cell ID SRXLEV - RX Level for Base Station [dB] RSRP - Serving Cell RSRP (Reference Signal Received Power [dBm]
X	Other modules	<Modulex>Module Payload in hexadecimal string format</Modulex>	

3.3.2.7 Modular Message (Message Type 11)

Byte	Description	TWPQueues	CellocatorHub
1	MCGP		Protocol = 1 (1= MCGP, 256 = CSA)
2			
3			
4			
5	Message Type (11)		MessageType = 11
6	Unit ID		CMUId
7			
8			
9			
10	Communication Control Field		MultiPurposeType Bits 0-1 (Bits 5-6)
11			Hibernation (Bit 7) FW Ver – Minor version (Bits 0-4)
12	Message Numerator (Anti-Tango™)		Numerator
13	Packet Control Field		
14	Length (of the modules section - not including the checksum)		
15			
16	0x0000 Symbolizes outbound message		
17			
18	Spare (sent as 0)		
19			
..	Modules payload		ModularData - see the following *Modular data table* section. Modules – a ';' separated list of module type ids.
Last Byte	Error Detection Code		

3.3.2.8 Modular data

Note that in TWPQueues this Modular Data for <Type11> messages is a separate table; in CellocatorHub it is in the "ModularData" column.

- Each module generates parsed relevant parameters in CellocatorHubDB.
- Additional data is transferred as XML.
- XMLs of all modules are surrounded by a <Type11> tag for CellocatorHub and with <PW_Uplink_CAN> tag for TWPQueues.
- Byte number in the following table is from the start of the module; Byte0 is the module ID, Byte1-2 are the module length, and Byte3 is the first byte of module data.

Id	Name	XML tag + CellocatorHub parameters	Remarks
1	DTC Code	<DTC> <I><M>3/7</M><V>Byte(5+2x)-(6+2x)</V></I> <I>…. </DTC>	Each DTC creates a <I> tag with <M> tag for mode 3/7 and <V> for DTC value (P/C/B/U + Code) where P-Power train, C-Chassis, B-Body, U-Network.
2	Trigger Event Module	<Trigger_Event> <OperatorID>Byte3-4</OperatorID> <PLSignature>Byte5-8</PLSignature> <Variables> <Variable VarID="VarID" Value="Value"/> </Variables> </Trigger_Event>	For each variable, a <Variable> tag is created with VarID and Value attributes in hexadecimal string format. The application should convert the value to units according to the var id and plSignature.
3	OBDII MIL Status	<OBD2MIL MilStat=Byte5Bit1/>	
4	Calibration Data Snap Shot	Odometer	@MileCounter
6	GPS Location Stamp Module	GPSMode1 GPSMode2 NumOfSat Lat Long Alt GPSSpeed GPSCourse	@GPSHDOP @GPSMode1 @GPSMode2 @SatelliteCount @GPSGeoX @GPSGeoY @GPSSpeed @GPSSpeedHeading @GPSCommStatus=1
7	GPS Time Stamp Module	UTCTime If Invalid time (Byte3=0): <Module7><TimeInvalid/></Module7>	@GPSLocationDateTime @UTCTime
8	Firmware ID Module	HWID FWVer	@VersionNum

Id	Name	XML tag + CellocatorHub parameters	Remarks
10	Configuration Memory Write Response	<MemWr Num="Byte3-4" Cnt="Byte5"> <I V="Byte6"> <I V="Byte7"> …. </MemWr>	Each block is an <I tag
11	Configuration Memory Read Response	<MemRd Num=Byte3-4 Cnt=Byte5> <I Typ="Type" Addr="Address" Lng="Length" V="ValueHexadecimal"> <I Typ="Type" Addr="Address" Lng="Length" V="ValueHexadecimal"> …. </MemRd>	Each block is an <I tag
12	CAN-GPS Calibration Status	<Module12> <CalbState>Byte5 </CalbState> <ConvMatrix>Byte6-7,Byte8-9…. </ConvMatrix> <RawData>Byte3-… </RawData> </Module12>	
20	Model Number	<ModelNumber> ValueAsUTF8string </ModelNumber>	
22	VIN String Write Module	<VIN>ValueAsUTF8string </VIN>	
28	Trigger Event ID Module	<Module28> <RawData>Byte3-… </RawData> </Module28> See the following *Module 28 XML table* section.	@TxReasonID @SourcePort
31	Variable Status Dump Module	<Trigger_Event_Vars> <Variable VarID="**VarID**" Value="**Value**"/> … </Trigger_Event_Vars>	For each variable a <Trigger_Event_Vars> tag is created with VarID and Value, the value is converted according to the format specified for each variable.
34	TPA - History Trip Event	<Module34> <TripID>Byte5-6 </TripID> <ZoneID>Byte7 </ZoneID> <ZoneFee>Byte8 </ZoneFee> <TpaTR>Byte9 </TpaTR> <Diag>Byte10 </Diag> <OdomKM>Byte11-13 </OdomKM> <TripDistKM>Byte14-16 </TripDistKM> <TripFee>Byte17-19</TripFee>	TripID History Trip ID ZoneID Complex Zone ID ZoneFee Complex Zone Fee TpaTR TPA Transmission Reason Diag Diagnostic bitmap OdomKM Vehicle Odometer (Km) TripDistKM Distance from History Trip Start TripFee Fee from History Trip Start

Id	Name	XML tag + CellocatorHub parameters	Remarks
		<ZoneDistKM>*Byte20-22* </ZoneDistKM> <LastFee>*Byte23-25* </LastFee> <Changes>*Byte26* </Changes> <RawData>*AllBytes*</RawData> </Module34>	ZoneDistKM Distance from Complex Zone Entrance or Fee change LastFee Fee from Complex Zone Entrance or Fee change Changes No of Zone/Fee changes in the Trip
37	TPA - Response to Request List of Sources reporting PGN 00FECA	<Module37> <Source No="*Byte4*"> <IndLamp PL="xx" AW="xx" RS="xx" MI="xx"/> <DTC> <I SPN="xx" FMI="xx" OC="xx" CM="xx"/> <I SPN="xx" FMI="xx" OC="xx" CM="xx"/> … </DTC> </Source> <RawData>*AllBytes*</RawData> </Module37>	Source No=" First Source Number IndLamp Physical Indication Lamp Status: PL Protect Lamp AW Amber warning lamp. RS Red stop lamp MI Malfunction Indicator For each DTC, an <I> tag is created: SPN – DTC Bits 0-18 FMI – DTC Bits 19-23 OC – DTC Bits 24-30 CM – DTC bit 31
38	OTA Single Source Status	<Module38> <TR_BM> byte4-5</TR_BM> <SrcNo> byte6</SrcNo> <IndLamp PL="xx" AW="xx" RS="xx" MI="xx"/> <DTC> <I SPN="xx" FMI="xx" OC="xx" CM="xx"/> <I SPN="xx" FMI="xx" OC="xx" CM="xx"/> … </DTC> <RawData>*AllBytes*</RawData> </Module38>	TR_BM Transmission Reasons Bitmask SrcNo Source Number IndLamp Physical Indication Lamp Status: PL Protect Lamp AW Amber warning lamp. RS Red stop lamp MI Malfunction Indicator For each DTC, an <I> tag is created: SPN – DTC Bits 0-18 FMI – DTC Bits 19-23 OC – DTC Bits 24-30 CM – DTC bit 31
40	Measurement readings	<Module40> <SrcEvnt>*byte10* </SrcEvnt> <NanoTxR>*byte8Bit0-2* </NanoTxR> <Status> *byte9Bit0-2* </Status> <End/> or <Start/> <Chrg> *byte9Bit4-5*</Chrg> <UpTH> *byte19*</UpTH> <Unk> / <Tmp> / <Hum> <I><V></V><T></T></I>	SrcEvnt – 0-15 Multisense, 0xFB-BT Extender, 0xFC – Guest Multisense, 0xFd - High accuracy or specialized sensors of the CelloTrack Nano 20, 0xFE- MCU internal NanoTxR - 0 – Normal 1 – Start Charging. 2 – Requested by Command. Status - Violation/Alert status 0 – Within the limits

Id	Name	XML tag + CellocatorHub parameters	Remarks
		<I><V></V><T></T></I> … </Unk> / </Tmp> / </Hum> <RawData>*AllBytes*</RawData> </ Module40> @SourcePort = byte10	1 – Within the limits 2 – Violating (not in alert) a lower TH but alert is not yet declared. 3 – Violating (not in alert) an upper TH but alert is not yet declared. 4 – Alert for lower TH violation 5 – Alert for upper TH violation 6 – Violating while in alert the lower TH. 7 – Violating while in alert the upper TH. <End/> or <Start/> will show according to byte9Bit3. Chrg Charging Status: 0 – Not charging* 1 – Charging slow. 2 – Charging fast. 3 – Charger Fault/Charger thermal shutdown UpTH/LoTH Upper/Lower threshold <Unk> / <Tmp> / <Hum> - section of samples Unknown / Temperature / Humidity <I><V></V><T></T></I> - item with Value <V> and date time <T>
41	Legacy message	<Module41> <RawData>*AllBytes*</RawData> </Module41> MessageType=11 HWID FWVer Hibernation GPSCommStatus HRNetwork PLMN TRSpecificData TRId UnitMode UTCTime GPSDateTime LastGPSFix	Parse all relevant data as message type 0.

Id	Name	XML tag + CellocatorHub parameters	Remarks
		Odometer	
42	Nano inherent sensors	<Module42> <XAcc>byte3-4</XAcc> <YAcc>byte5-6</YAcc> <ZAcc>byte7-8</ZAcc> <Lgt>byte9-10</Lgt> <Alt>byte11-12</Alt> <Tmp>byte13-14</Tmp> <Bat>byte16</Bat> <LRSSI>byte17</LRSSI> <RawData>AllBytes</RawData> <InvalidTmp/> <PositionUpright/>/<PositionLaying/> </Module42>	XAcc/YAcc/ZAcc – acceleration (g) Lgt – Light (lux) Alt Average of 2 last Air pressure samples, translated to altitude above sea-level. Tmp Current temperature Bat - Battery level (%) LRSSI - Last RSSI value (dBM) <InvalidTmp/>,<PositionUpright/>/<PositionLaying/> - are optional tags that are added according to byte 15 flags.
44	MultiSense additional info	<Module44> <Mac>byte3-8</XAcc> <Bat>byte9</Bat> <RSSI>byte10</RSSI> <LastComm>byte11-16</LastComm> <MSFW>byte17-18</MSFW> <BOMMask>byt19-20</BOMMask> <Tmp>byte21-22</Tmp> <Hum>byte23-24</Hum> <Lgt>byte25-26</Lgt> <XAcc>byte27-28</XAcc> <YAcc>byte29-30</YAcc> <ZAcc>byte31-32</ZAcc> <Mgnt> byte33bit0</Mgnt> <Pkg> byte33bit1</Pkg> <AccSTst>byte33bit7</AccSTst> <RawData>AllBytes</RawData> </Module44>	MAC – MAC address (hexadecimal) Bat - Battery level (%) RSSI - RSSI (Signed, dBm units) LastComm - Time of last communication from the specific MultiSense MSFW – Multisense FW BOMMask - Bom mask bitmap Tmp - Last measured temperature Hum – Humidity (%) Lgt – Light (lux) XAcc/YAcc/ZAcc – acceleration (g) Mgnt - Magnetic Sensor True/False Pkg – package (open/close) AccSTst - Accelerometer Self (Pass/Fail) Tmp, Hum, Lgt, XAcc/YAcc/ZAcc are optional according to BOMMask
45	Entire system MultiSense readings	<Module45> <TxR>byte4</TxR> <Idx>byte5</Idx> <Mac>byte6-11</XAcc> <MsTx>byte12</MsTx> <ThSt>byte13</ThSt> <Bat>byte14</Bat> <RSSI>byte15</RSSI>	TxR TX Reason 0 – Retransmission 1 – Not retransmission 2 – A violating area, with violation sampling rate. Idx MultiSense index in the system (0-15) MAC – MAC address (hexadecimal)

Id	Name	XML tag + CellocatorHub parameters	Remarks
		<Prtc>byte16</Prtc> <MSFW>byte17-18</MSFW> <BOMMask>byt19-20</BOMMask> <Tmp>byte21-22</Tmp> <Hum>byte23-24</Hum> <Lgt>byte25-26</Lgt> <XAcc>byte27-28</XAcc> <YAcc>byte29-30</YAcc> <ZAcc>byte31-32</ZAcc> <Mgnt> byte33bit0</Mgnt> <Pkg> byte33bit1</Pkg> <AccSTst>byte33bit7</AccSTst> <RawData>AllBytes</RawData> </Module45> @SourcePort = byte5 – if not retransmission. @TxReasonId = 1164 @AlertReason = 139 - Temperature measurements violation 151 - Humidity/Temperature measurements violation 12 - Humidity measurements violation	MsTx Last MultiSense TX reason ThSt TH Status bitmap Bat Battery level (%) RSSI RSSI (Signed, dBm units) Prtc OTA protocol version MSFW MultiSense FW Version BOMMask Bom mask bitmap Tmp Last measured temperature Hum Humidity (%) Lgt Light (lux) XAcc/YAcc/ZAcc acceleration (g) Mgnt Magnetic Sensor True/False Pkg package (open/close) AccSTst Accelerometer Self (Pass/Fail)
46	3G/4G Cell-ID	<Module46> <Cell> <CellType>4G</CellType> <Time>Byte4-6</Time> <Date>Byte7-9</ Date > <MCC>Byte10-11</MCC> <MNC>Byte12-13</MNC> <TAC>Byte14-15</TAC> <GCellID>Byte17-22</GCellID> <CellID>Byte23-24</CellID> <RSRP>Byte25</RSRP> <AcT>Byte26</AcT> <Neighbors> <Nx> <MCC>Byte30-31</MCC> <MNC>…</MNC> <TAC>…</TAC> <GCellID>…</GCellID>	

Id	Name	XML tag + CellocatorHub parameters	Remarks
		<CellID>…</CellID> <RSRP>… </RSRP> <AcT>… </AcT> </Nx> <Ny … </Neighbors> </Cell> <RawData>*AllBytes*</RawData> </Module46>	
49	Data logger compressed block	<Module49> <Msr> <Tmp> <I><V></V><T></T></I> <I><V></V><T></T></I> … </Tmp> <Hum> <I><V></V><T></T></I> … </Hum> </Msr> <Evn> <I> <V></V><T></T> <TxR></TxR><ThSt></ThSt> <AccX></AccX><AccY></AccY><AccZ> </AccZ><Bat></Bat> </I> <I>… </Evn> <RawData>*AllBytes*</RawData> </Module49>	temperature <Tmp> or Humidity <Hum> tags contain items <I><V></V><T></T></I> - with Value <V> and date time <T> Events <Evn> tags contain items <I><V></V><T></T></I> - with Value <V> , date time <T>, <TxR> TX Reason 0 – Retransmission 1 – Not retransmission 2 – A violating area, with violation sampling rate <ThSt> TH Status bitmap <XAcc>/<YAcc>/<ZAcc> – acceleration (g) <Bat>-Battery level (%)
50	DTCO Connect/Disconnect Event	Supported - beta	
51	DTCO Time Stamp	Supported - beta	
52	DTCO Driver Identification Numbers	Supported - beta	

Id	Name	XML tag + CellocatorHub parameters	Remarks
53	DTCO Parameters Change Event	Supported - beta	
54	DTCO Periodic Event	Supported - beta	
60	TDLT Event (OneLink)	</Module60> <IMEI>*byte3-9*</IMEI> <LgIn>*Byte12Bit0*</LgIn> <CrdVld>*Byte12Bit2*</CrdVld> <VclTyp>*AllBytes*</VclTyp> <RSSI>*byte22*</RSSI> <XAcc>*byte23-24*</XAcc> <YAcc>*byte25-26*</YAcc> <ZAcc>*byte27-28*</ZAcc> <Crd1>*byte33*</Crd1> <Crd2>…</Crd2> <Crd3>…</Crd3> <RawData>*AllBytes*</RawData> </Module60>	IMEI string LgIn Login Status 0 – Not Logged In 1 – Logged In. CrdVld Last Card Status 0 – Not Valid 1 – Valid VclTyp Vehicle Type RSSI - RSSI (Signed, dBm units) XAcc/YAcc/ZAcc – acceleration (g) Crd1/Crd2/Crd3 Magnetic Card Track #1/#2/#3
90		<Module90> <Bat>*byte5*</Bat> <Status>*byte6-7*</Status> <Tmp>*byte10-11*</Tmp> <RSSI>byte14</RSSI> <RawData>*AllBytes*</RawData> </Module90>	Bat- Battery State of Health (%) Status – 16bits Bitmap of Battery Fuel Gauge Status Tmp - Temperature Reading degrees RSSI - RSSI (Signed, dBm units)
	Other modules	<Module??> <RawData>*AllBytes*</RawData> </Module??>	The ?? represents the module ID; each unfamiliar module is parsed to a generic XML template.

3.3.2.9 *Module 28 XML table*

Event Category (Byte4-Byte5Bit6)	Event Code (Byte 6-7)	XML / CellocatorHub / TWPQueues	Remarks
0	1	<Code>byte9</Code> <Data1>byte10-11</Data1> <Data2>byte12-13</Data2>	OBD2 Query Event
0	2	<SubCode>byte10</SubCode>	CAN BUS Event
1,2,3	1,16	<SrcEvnt>byte9</SrcEvnt> <AccRMS>byte10-13</AccRMS>	Impact/Freefall
1,2,3	2,3	@TxReasonId=2/3 @SourcePort=0xfd	Orientation change/Man down
1,2,3	4	<SrcEvnt>byte9 </SrcEvnt> <IsOpen>byte10 </IsOpen> <Lgt>byte11-12</Lgt> @SourcePort=byte9	Light sensor
1,2,3	7	<MSDataLogger/> <SrcEvnt>byte9</SrcEvnt> <PrbCode>byte10</PrbCode> <Bat>byte11</Bat> <LRSSI>byte12</LRSSI> <SystemTime>byte13-18</SystemTime> <MltSnsFW>byte19-20</MltSnsFW> <BOMMask>byte20-21</BOMMask> <Tmp>byte23-24</Tmp> <Hum>byte25-26</Hum> <Lgt>byte27-28</Lgt> <XAcc>byte29-30</XAcc> <YAcc>byte31-32</YAcc> <ZAcc>byte33-34</ZAcc> <Mgnt>byte35Bit0</Mgnt> <Pkg>Bute35Bit</Pkg> <AccSTst>Byte35Bit7</AccSTst> @SourcePort=byte9	MultiSense provisioning message MSDataLogger – exists when the source is MultiSense Data Logger PrbCode - Problem Code Bat - Battery level (%) LRSSI- Last RSSI value (dBM) MltSnsFW - MultiSense FW version BOMMask - Bom mask bitmap Tmp - Last measured temperature Hum – Humidity (%) Lgt – Light (lux) XAcc/YAcc/ZAcc – acceleration (g) Mgnt - Magnetic Sensor True/False Pkg – Package (open/close) AccSTst - Accelerometer Self (Pass/Fail)

Event Category (Byte4-Byte5Bit6)	Event Code (Byte 6-7)	XML / CellocatorHub / TWPQueues	Remarks
1,2,3	8,9	<SrcEvnt>*byte9*</SrcEvnt> <Act>*Byte0* </Act> <MAC>*byte10-15*</MAC> <BOMMask>*byte16*</BOMMask> <Bat>*byte17*</SystemTime> <LRSSI>*byte18*</LRSSI> @SourcePort=byte9	MultiSense added/removed. Act – Action (Add/Remove) MAC – MAC address (hexadecimal) BOMMask - Bom mask bitmap Bat - Battery level (%) LRSSI - Last RSSI value(dBM)
1,2,3	10	<WrkID>*byte9-13*</WrkID> @SourcePort=0xFD	Work-ID / Activation WrkID – worker ID as 32bit unsigned int.
1,2,3	11	@TxReasonId=11 @SourcePort=0xFD	Check-in
1,2,3	13	<RSSI>*byte10*</RSSI> <MAC>*byte11-16*</MAC> <OTAVer>*byte17*</OTAVer> <Bat>*byte18-19*</Bat> <BOMMask>*byte20-21*</BOMMask> <TR>*byte22*</TR> <FWVer>*byte23-24*</FWVer> <Tmp>*byte26-27*</Tmp> <Hum>*byte28-29*</Hum> <Lgt>*byte30-31*</Lgt> <Mgnt>*byte35Bit0*</Mgnt> <XAcc>*byte26-27*</XAcc> <YAcc>*byte28-29*</YAcc> <ZAcc>*byte30-31*</ZAcc> <AccSTst>*Byte32Bit7*</AccSTst> <Pkg>*byte32Bit1*</Pkg> <THState>*byte41*</THState> <RawAdv>*byte10-25*</RawAdv> <RawSnsRead>*byte26-42*</RawSnsRead> @SourcePort=byte9	MultiSense provisioning message RSSI - RSSI (Signed, dBm units) MAC – MAC address (hexadecimal) OTAVer - OTA Protocol Version Bat - Battery level (%) BOMMask - Bom mask bitmap TR – Transmission reason FWVer - MultiSense FW Version Tmp - Last measured temperature Hum – Humidity (%) Lgt – Light (lux) Mgnt - Magnetic Sensor True/False XAcc/YAcc/ZAcc – acceleration (g) AccSTst - Accelerometer Self (Pass/Fail) Pkg – Package (open/close) THState – Threshold state bitmap RawAdv & RawSnsRead – Raw data in hexadecimal format. Tmp, Hum, Lgt, Mgnt XAcc/YAcc/ZAcc are optional according to BOMMask

Event Category (Byte4-Byte5Bit6)	Event Code (Byte 6-7)	XML / CellocatorHub / TWPQueues	Remarks
1,2,3	15	<SrcEvnt>*byte9*</SrcEvnt> <State>*byte10*</State> @SourcePort=byte9	Door/window State – Close/Open
1,2,3	17	<RSSI>*byte10*</RSSI> <MAC>*byte11-16*</MAC> <OTAVer>*byte17*</OTAVer> <Bat>*byte18-19*</Bat> <BOMMask>*byte20-21*</BOMMask> <TR>*byte22*</TR> <FWVer>*byte23-24*</FWVer> <RawAdv>*byte10-25*</RawAdv> @SourcePort=byte9	Tag mode MultiSense raw data RSSI - RSSI (Signed, dBm units) MAC – MAC address (hexadecimal) OTAVer - OTA Protocol Version Bat - Battery level (%) BOMMask - Bom mask bitmap TR – Transmission reason FWVer - MultiSense FW Version RawAdv – Raw data in hexadecimal format
1,2,3	18	<SrcEvnt>*byte9*</SrcEvnt> <MDif>*byte10-13*</MDif> SourcePort=byte9	Sudden pressure change MDif - Difference between the last stable altitude and current filtered one
1,2,3	21	<SrcEvnt>*byte9*</SrcEvnt> <SecSinceLastTrans>*byte10-13*</SecSinceLastTrans> SourcePort=byte9	No MultiSense received SecSinceLastTrans – Time elapsed from last transmission [Seconds]
1,2,3	23	<SrcEvnt>*byte9*</SrcEvnt> SourcePort=byte9	
1,2,3	24	<SrcEvnt>*byte9*</SrcEvnt> <PrbCode>*byte10*</PrbCode> <BTMAC>*byte11-16*</BTMAC> <BTName>*byte17..*</BTName> SourcePort=byte9	BT classic connected/disconnected. PrbCode - Problem Code 0: Reserved, 1: Connected, 2: Disconnected BTMAC - BT master (host) MAC address BTName - BT classic master friendly name

Event Category (Byte4-Byte5Bit6)	Event Code (Byte 6-7)	XML / CellocatorHub / TWPQueues	Remarks
1,2,3	25	<SrcEvnt>*byte9*</SrcEvnt> <PrbCode>*byte10*</PrbCode> <MdlID>*byte11*</BTMdlIDMAC> <HWRev>*byte12-13*</BTName> <FWBL>*byte14-15*</FWBL> <FWMj>*byte16-17*</FWMj> <FWMi>*byte18-19*</FWMi> <FWPt>*byte20-21*</FWPt> <FWBu>*byte22-23*</FWBu> SourcePort=byte9	BT-Extender connected/disconnected. PrbCode - Problem code 0: Reserved 1: Lost communication 2: Communication restored. MdlID - Module ID HWRev - HW Revision FWBL - FW Version – Bootloader FWMj - FW Version – Major FWMi - FW Version – Minor FWPt - FW Version – Patch FWBu - FW Version – Build

3.3.3 *CSA Messages*

Note the following:

- All CSA messages are sent as XML in ModularData (as shown in the following table).
- XML tag <NXS02XXX> includes:
  - <Info> tag with message and unit information.
  - <NXS02030> tag includes <Count> - number of modules, and module data in the <Module> tag.
  - <NXS02031> tag includes <CsaData> that includes the raw data.

3.3.3.1 Modules

Id	Name	XML tag + CellocatorHub parameters	XML
0	Event Info	TRId = EventReason TRSpecificData = EventSubReason Numerator = EventNumerator DrivingStatus = OperationMode Bit4 IgnitionPhysical = OperationMode Bit7	<Module><ID>0</ID><Data><MMEventInfo><EventReason>0</EventReason><EventSubReason>0</EventSubReason><EventNumerator>0</EventNumerator><OperationMode>0</OperationMode></MMEventInfo></Data></Module>
1	Driver ID	DriverID = DriverID	<Module><ID>1</ID><Data><MMDriverID><DriverID>0</DriverID></MMDriverID></Data></Module>
2	TripID	TripId = TripID	<Module><ID>2</ID><Data><MMTripID><TripID>0</TripID></MMTripID></Data></Module>
3	Maneuver ID	ManeuverId =ManeuverID	<Module><ID>3</ID><Data><MMManeuverID><ManeuverID>0</ManeuverID></MMManeuverID></Data></Module>
4	PLMN	PLMN = PLMN	<Module><ID>4</ID><Data><MMPLMN><PLMN>0</PLMN></MMPLMN></Data></Module>
5	Memory Usage Status		<Module><ID>35</ID><Data><MMCrashAttributes><CrashID>0</CrashID><VehicleType>Private</VehicleType><CrashType>Light_Crash</CrashType><MaxG>0</MaxG><CrashInfo>INFO_INIT</CrashInfo><CrashRoll>Roll_Event</CrashRoll><CrashParking>Driving</CrashParking><DurationInSeconds>0</DurationInSeconds><NumOfInitAccel>0</NumOfInitAccel><amoutOfPostSamples>0</amoutOfPostSamples><PreDataLenInSec>0</PreDataLenInSec><PostDataLenInSec>0</PostDataLenInSec></MMCrashAttributes></Data></Module>

Id	Name	XML tag + CellocatorHub parameters	XML
6	GPS Stamp	GPSMode1= Mode1 GPSMode2 = Mode2; NumOfSat =NumberOfSatellitesUsed Long = Longitude Lat = Latitude Alt = Altitude GPSSpeed = GroundSpeed GPSCourse = SpeedDirection	<Module><ID>6</ID><Data><MMGpsStamp><HDOP>0</HDOP><Mode1>0</Mode1><Mode2>0</Mode2><NumberOfSatellitesUsed>0</NumberOfSatellitesUsed><Longitude>0</Longitude><Latitude>0</Latitude><Altitude>0</Altitude><GroundSpeed>0</GroundSpeed><SpeedDirection>0</SpeedDirection></MMGpsStamp></Data></Module>
7	Time Stamp	UTCTime = Year+Month+Day Hour+Minute+Second If "ValiditiyOfTime" == 1 tjem GPSDateTime=UTCTime	<Module><ID>7</ID><Data><MMTimeStamp><ValidityOfTime>0</ValidityOfTime><Seconds>9</Seconds><Minutes>40</Minutes><Hours>15</Hours><Day>17</Day><Month>12</Month><Year>18</Year></MMTimeStamp></Data></Module>
8	CSA FW ID	FWVer	<Module><ID>8</ID><Data><MMCsaFwID><ProtocolID>4</ProtocolID><CSA_FW_ID>131073</CSA_FW_ID><HW_Type>2</HW_Type><UnitType>CELLO_IQ50</UnitType><ProductionID>0</ProductionID></MMCsaFwID></Data></Module>
30	Full Event	TRId = EventReason TRSpecificData = EventSubReason Numerator = EventNumerator DrivingStatus = OperationMode Bit4 IgnitionPhysical = OperationMode Bit7 DriverID = DriverID TripId = TripID ManeuverId =ManeuverID GPSMode1= Mode1 GPSMode2 = Mode2; NumOfSat =NumberOfSatellitesUsed Long = Longitude Lat = Latitude Alt = Altitude	<Module><ID>30</ID><Data><MMFullEvent><EventReason>0</EventReason><EventSubReason>0</EventSubReason><EventNumerator>0</EventNumerator><OperationMode>0</OperationMode><StandbyEngineFlag>Off</StandbyEngineFlag><DFDCommunicationFlag>Connected</DFDCommunicationFlag><DrivingFlag>Idling</DrivingFlag><CalibratingFlag>Ready</CalibratingFlag><RawLoggingFlag>Off</RawLoggingFlag><EngineOnFlag>Off</EngineOnFlag><DriverID>0</DriverID><TripID>0</TripID><ManeuverID>0</ManeuverID><ManeuversDataUsage>0</ManeuversDataUsage><AccidentBufferStatusBitmak>0</AccidentBufferStatusBitmak><AccidentBufferStatusBitmaskCrash1>Empty</AccidentBufferStatusBitmaskCrash1><AccidentBufferStatusBitmaskCrash2>Empty</AccidentBufferStatusBitmaskCrash2><HDOP>0</HDOP><Mode1>0</Mode1><Mode2>0</Mode2><SatellitesUsed>0</SatellitesUsed><Longitude>0</Longitude><Latitude>0</Latitude><Altitude>0</Altitude><GroundSpeed>0</GroundSpeed><SpeedDirection>0</SpeedDirection><Seconds>16</Seconds><Minutes>40</Minutes><Hours>15</Hours><Day>17</Day><Month>12</Month><Year>18</Year></MMFullEvent></Data></Module>

Id	Name	XML tag + CellocatorHub parameters	XML
		GPSSpeed = GroundSpeed GPSCourse = SpeedDirection UTCTime = Year+Month+Day Hour+Minute+Second	
31	Maneuver Statistics	TripId = TripID ManeuverId = ManeuverID	<Module><ID>31</ID><Data><MMManeuverStatistics><TripID>0</TripID><ManeuverID>0</ManeuverID><ManeuverType>Crash_Occurred</ManeuverType><StartLocation>0</StartLocation><EndLocation>0</EndLocation><StartTime>0</StartTime><ManeuverDuration>0</ManeuverDuration><XAverage>0</XAverage><YAverage>0</YAverage><MaxX>0</MaxX><MaxY>0</MaxY><MaxZ>0</MaxZ><SpeedAverage>0</SpeedAverage><SpeedMax>0</SpeedMax><SpeedDelta>0</SpeedDelta><MaxRPM>0</MaxRPM><MaxFuelFlow>0</MaxFuelFlow><FuelConsumed>0</FuelConsumed><ABSState>0</ABSState><RiskScore>0</RiskScore><NumOfInitFrames>0</NumOfInitFrames></MMManeuverStatistics></Data></Module>
32	Trip Statistics	TripId = TripID DriverID = DriverID	<Module><ID>32</ID><Data><MMTripStatistics><TripID>0</TripID><DriverID>0</DriverID><StartDateTime>0</StartDateTime><TripDurationSeconds>0</TripDurationSeconds><DistanceTraveled>0</DistanceTraveled><MovementTime>0</MovementTime><IdleTimeShort>0</IdleTimeShort><IdleTimeLong>0</IdleTimeLong><MaxX>0</MaxX><MaxY>0</MaxY><MaxZ>0</MaxZ><MaxRMS>0</MaxRMS><MaxSpeed1>0</MaxSpeed1><AVGSpeed>0</AVGSpeed><StartFuelLevel>0</StartFuelLevel><EndFuelLevel>0</EndFuelLevel><WeightedSafetyScore>0</WeightedSafetyScore><WeightedEcoScore>0</WeightedEcoScore><EcoScoringIdleScore>0</EcoScoringIdleScore><EcoScoringUrbanDrivingScore>0</EcoScoringUrbanDrivingScore><EcoScoringHighwayDrivingScore>0</EcoScoringHighwayDrivingScore><EcoScoringIdleScoreTime>0</EcoScoringIdleScoreTime><EcoScoringUrbanScoreTime>0</EcoScoringUrbanScoreTime><EcoScoringHighwayScoreTime>0</EcoScoringHighwayScoreTime><TripEndOdometer>0</TripEndOdometer><DeltaEngineHours>0</DeltaEngineHours><CurrentTripMovementFuelConsumption>0</CurrentTripMovementFuelConsumption><CurrentTripIdleFuelConsumption>0</CurrentTripIdleFuelConsumption></MMTripStatistics></Data></Module>

Id	Name	XML tag + CellocatorHub parameters	XML
34	Trip Log	TripId = TripID	<Module><ID>34</ID><Data><MMTripLog><TripID>0</TripID><NumberOfManeuvers>10</NumberOfManeuvers><Table><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort><ArrayOfUnsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort><unsignedShort>0</unsignedShort></ArrayOfUnsignedShort></Table></MMTripLog></Data></Module>
35	EDR Crash Attributes		<Module><ID>35</ID><Data><MMCrashAttributes><CrashID>0</CrashID><VehicleType>Private</VehicleType><CrashType>Light_Crash</CrashType><MaxG>0</MaxG><CrashInfo>INFO_INIT</CrashInfo><CrashRoll>Roll_Event</CrashRoll><CrashParking>Driving</CrashParking><DurationInSeconds>0</DurationInSeconds><NumOfInitAccel>0</NumOfInitAccel><amoutOfPostSamples>0</amoutOfPostSamples><PreDataLenInSec>0</PreDataLenInSec><PostDataLenInSec>0</PostDataLenInSec></MMCrashAttributes></Data></Module>
37	Calibration/ Off Road Status		<Module><ID>37</ID><Data><MMCalibrationStatus><CalibrationStatus>Calibration</CalibrationStatus><CalibrationStep>1</CalibrationStep><CalibrationPhaze>2</CalibrationPhaze><OffroadStage>Not_calibrated</OffroadStage></MMCalibrationStatus></Data></Module></NXS02030><NXS02031><CsaData>4353410D001000C80100000025030021002F</CsaData></NXS02031></NXS02XXX>

Id	Name	XML tag + CellocatorHub parameters	XML
53	Off Road Setup Data Response		<Module><ID>53</ID><Data><MMResponseOffroadSetupDataSetCommand><Index><int>1</int><int>2</int><int>0</int><int>100</int><int>0</int><int>0</int><int>0</int><int>0</int><int>0</int><int>0</int><int>0</int><int>0</int><int>0</int><int>0</int><int>0</int><int>1</int><int>1</int><int>0</int><int>1</int><int>1</int><int>21</int></Index><Spare>010200000005</Spare></MMResponseOffroadSetupDataSetCommand></Data></Module></NXS02030><NXS02031><CsaData>43534164000300C801000000355A010000000200000000000000640000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000100000000000000001000000010000001500000000102000000000547</CsaData>
55	Real Time Raw Accelerometer Data		<NXS02XXX xmlns="http://nexus.co.il/NXS02XXX.xsd"><Info><Ver>3</Ver><MsgId>17</MsgId><Time>19/12/2018 09:37:42</Time><Status>0</Status><UnitId>1</UnitId><SrcTypeId>0</SrcTypeId><SrcId>1</SrcId></Info><NXS02030><Count>1</Count><Module><ID>55</ID><Data><MMRealTimeRawAccData><SampleSize>_1Byte</SampleSize><NumberOfIncludedSamples>0</NumberOfIncludedSamples><SamplingFrequency>_100Hz</SamplingFrequency><FrameNumber>0</FrameNumber><IsLastFrame>false</IsLastFrame><AccurateTime>Estimated</AccurateTime><CalibrationStatus>Not_Calibrated</CalibrationStatus><AccelerometerResolution>_8bits</AccelerometerResolution><Seconds>0</Seconds><Minutes>0</Minutes><Hours>0</Hours><Day>0</Day><Month>0</Month><Year>0</Year><Samples /></MMRealTimeRawAccData></Data></Module>
56	Restore Default Vehicle Preset		<Module><ID>56</ID><Data><MMRestorePresetCommandRespond><VehicleType>Private</VehicleType><Spare>0</Spare></MMRestorePresetCommandRespond></Data></Module>
57	Vehicle Preset Configuration Command		Module><ID>57</ID><Data><MMVehiclePresetConfiguration><ControlByte v="Read_Command"/><VehicleType v="Private"/><SystemConfigurationData v="0000000000000000000"/><FrameSize v="0"/><XMax v="0"/><Spare v="0"/><Filter v="0"/><AmountOfViolatingSamplesToConsiderCrash v="0"/><AccelerationOnXAxisToConsiderManeuverStart v="0"/><turnYThreshold v="0"/><AccelerationOnXAxisToConsiderComboManeuverStart v="0"/><turnSecondYThreshold v="0"/><TimeThresholdForConsideringLaneCrossing v="0"/><TimeThresholdForConsideringManeuver v="0"/><breakXThreshold v="0"/><breaksecondXThreshold

Id	Name	XML tag + CellocatorHub parameters	XML
			v="0"/><MaximumManeuverLength v="0"/><AccelerationXavg v="0"/><AccelerationXmax v="0"/><AccelerationDeltaSpeed v="0"/><AccelerationSpare v="0"/><ACCEL2TURNAccelXavg v="0"/><ACCEL2TURNAccelXmax v="0"/><ACCEL2TURNDeltaSpeed v="0"/><ACCEL2TURNYavg v="0"/><ACCEL2TURNYmax v="0"/><ACCEL2TURNAvgSpeed v="0"/><TURN_AFTER_ACCELXavg v="0"/><TURN_AFTER_ACCELXmax v="0"/><TURN_AFTER_ACCELDeltaSpeed v="0"/><TURN_AFTER_ACCELYavg v="0"/><TURN_AFTER_ACCELYmax v="0"/><TURN_AFTER_ACCELAvgSpeed v="0"/><SLALOM_AFTER_ACCELXavg v="0"/><SLALOM_AFTER_ACCELXmax v="0"/><SLALOM_AFTER_ACCELDeltaSpeed v="0"/><SLALOM_AFTER_ACCELSLALOMYavg v="0"/><SLALOM_AFTER_ACCELSLALOMYmax v="0"/><SLALOM_AFTER_ACCELSLALOMSpeedMax v="0"/><ACCEL2SLALOM_AAccelXavg v="0"/><ACCEL2SLALOM_AAccelXmax v="0"/><ACCEL2SLALOM_AAccelDeltaSpeed v="0"/><ACCEL2SLALOM_ASLALOMYavg v="0"/><ACCEL2SLALOM_ASLALOMYmax v="0"/><ACCEL2SLALOM_ASLALOMSpeedMax v="0"/><ACCEL2SLALOM_TAccelXavg v="0"/><ACCEL2SLALOM_TAccelXmax v="0"/><ACCEL2SLALOM_TAccelDeltaSpeed v="0"/><ACCEL2SLALOM_TSLALOMYavg v="0"/><ACCEL2SLALOM_TSLALOMYmax v="0"/><ACCEL2SLALOM_TSLALOMSpeedMax v="0"/><BRAKEXavg v="0"/><BRAKEXmax v="0"/><BRAKEDeltaSpeed v="0"/><BRAKESpare v="0"/><BRAKE2TURNBrakeXavg v="0"/><BRAKE2TURNBrakXmax v="0"/><BRAKE2TURNBrakeDeltaSpeed v="0"/><BRAKE2TURNTURNYavg v="0"/><BRAKE2TURNTURNYmax v="0"/><BRAKE2TURNTURNSpeedAvg v="0"/><TURN_AFTER_BREAKBrakeXavg v="0"/><TURN_AFTER_BREAKBrakXmax v="0"/><TURN_AFTER_BREAKBrakeDeltaSpeed v="0"/><TURN_AFTER_BREAKTURNYavg v="0"/><TURN_AFTER_BREAKTURNYmax v="0"/><TURN_AFTER_BREAKTURNSpeedAvg v="0"/><SLALOM_AFTER_BRAKEBrakeXavg v="0"/><SLALOM_AFTER_BRAKEBrakeXmax v="0"/><SLALOM_AFTER_BRAKEBrakeDeltaSpeed v="0"/><SLALOM_AFTER_BRAKESLALOMYavg v="0"/><SLALOM_AFTER_BRAKESLALOMYmax v="0"/><SLALOM_AFTER_BRAKESLALOMSpeedMax v="0"/><BRAKE2SLALOMBBrakeXavg v="0"/><BRAKE2SLALOMBBrakeXmax v="0"/><BRAKE2SLALOMBBrakeDeltaSpeed v="0"/><BRAKE2SLALOMBSLALOMYavg v="0"/><BRAKE2SLALOMBSLALOMYmax

Id	Name	XML tag + CellocatorHub parameters	XML
			v="0"/><BRAKE2SLALOMBSLALOMSpeedMax v="0"/><BRAKE2SLALOMTBrakeXavg v="0"/><BRAKE2SLALOMTBrakeXmax v="0"/><BRAKE2SLALOMTBrakeDeltaSpeed v="0"/><BRAKE2SLALOMTSLALOMYavg v="0"/><BRAKE2SLALOMTSLALOMYmax v="0"/><BRAKE2SLALOMTSLALOMSpeedMax v="0"/><TURNYavg v="0"/><TURNYmax v="0"/><TURNSpeedAvg v="0"/><TURNSpare1 v="0"/><TURNSpare2 v="0"/><TURNSpare3 v="0"/><TURN2BRAKETurnYavg v="0"/><TURN2BRAKETurnYmax v="0"/><TURN2BRAKETurnSpeedAvg v="0"/><TURN2BRAKEBrakeXavg v="0"/><TURN2BRAKEBrakeXmax v="0"/><TURN2BRAKEBrakeDeltaSpeed v="0"/><BRAKE_AFTER_TURNTurnYavg v="0"/><BRAKE_AFTER_TURNTurnYmax v="0"/><BRAKE_AFTER_TURNTurnSpeedAvg v="0"/><BRAKE_AFTER_TURNBrakeXavg v="0"/><BRAKE_AFTER_TURNBrakeXmax v="0"/><BRAKE_AFTER_TURNBrakeDeltaSpeed v="0"/><TURN2ACCELTurnYavg v="0"/><TURN2ACCELTurnYmax v="0"/><TURN2ACCELTurnSpeedAvg v="0"/><TURN2ACCELAccelXavg v="0"/><TURN2ACCELAccelXmax v="0"/><TURN2ACCELAccelDeltaSpeed v="0"/><ACCEL_AFTER_TURNTurnYavg v="0"/><ACCEL_AFTER_TURNTurnYmax v="0"/><ACCEL_AFTER_TURNTurnSpeedAvg v="0"/><ACCEL_AFTER_TURNAccelXavg v="0"/><ACCEL_AFTER_TURNAccelXmax v="0"/><ACCEL_AFTER_TURNAccelDeltaSpeed v="0"/><SLALOMYavg v="0"/><SLALOMYmax v="0"/><SLALOMSpeedMax v="0"/><SLALOMSpare1 v="0"/><SLALOMSpare2 v="0"/><SLALOMSpare3 v="0"/><SLALOM2BRAKE_BSlalomYmax v="0"/><SLALOM2BRAKE_BSlalomYavg v="0"/><SLALOM2BRAKE_BSlalomSpeedMax v="0"/><SLALOM2BRAKE_BBrakeXavg v="0"/><SLALOM2BRAKE_BBrakeXmax v="0"/><SLALOM2BRAKE_BBrakeDeltaSpeed v="0"/><SLALOM2BRAKE_TSlalomYmax v="0"/><SLALOM2BRAKE_TSlalomYavg v="0"/><SLALOM2BRAKE_TSlalomSpeedMax v="0"/><SLALOM2BRAKE_TBrakeXavg v="0"/><SLALOM2BRAKE_TBrakeXmax v="0"/><SLALOM2BRAKE_TBrakeDeltaSpeed v="0"/><SLALOM2ACCEL_ASlalomYmax v="0"/><SLALOM2ACCEL_ASlalomYavg v="0"/><SLALOM2ACCEL_ASlalomSpeedMax v="0"/><SLALOM2ACCEL_AAccelXavg v="0"/><SLALOM2ACCEL_AAccelXmax v="0"/><SLALOM2ACCEL_AAccelDeltaSpeed v="0"/><SLALOM2ACCEL_TSlalomYmax v="0"/><SLALOM2ACCEL_TSlalomYavg v="0"/><SLALOM2ACCEL_TSlalomSpeedMax v="0"/><SLALOM2ACCEL_TAccelXavg

Id	Name	XML tag + CellocatorHub parameters	XML
			v="0"/><SLALOM2ACCEL_TAccelXmax v="0"/><SLALOM2ACCEL_TAccelDeltaSpeed v="0"/><BUMPERS v="000000000000"/><SPEEDING v="000000000000"/><OFF_ROAD v="000000000000"/><IDLE_Event v="000000000000"/><STATES_RPM v="000000000000"/><CRASH v="000000000000"/><WeightACCELXavg v="0"/><WeightACCELXmax v="0"/><WeightACCELDeltaspeed v="0"/><WeightBRAKEXavg v="0"/><WeightBRAKEXmax v="0"/><WeightBRAKEDeltaspeed v="0"/><WeightTURNYavg v="0"/><WeightTURNYmax v="0"/><WeightTURNAvgspeed v="0"/><WeightSLALOMYavg v="0"/><WeightSLALOMYmax v="0"/><WeightSLALOMSpeedmax v="0"/><StatusBarThreshold1 v="0"/><StatusBarThreshold2 v="0"/><StatusBarThreshold3 v="0"/><StatusBarThreshold4 v="0"/><EcoRpmGreen1 v="0"/><EcoRpmYellow1 v="0"/><EcoRpmRed1 v="0"/><EcoScore1 v="0"/><EcoScore2 v="0"/><EcoScore3 v="0"/><EcoScore4 v="0"/><EcoScore5 v="0"/><EcoScore6 v="0"/><EcoSpeed1 v="0"/><EcoSpeed2 v="0"/><EcoSpeed3 v="0"/><EcoSpeed4 v="0"/><EcoSpeed5 v="0"/><EcoSpeed6 v="0"/><RPMWeight v="0"/><SpeedWeight v="0"/><Spare1 v="0"/><Spare2 v="0"/></MMVehiclePresetConfiguration></Data></Module></NXS02030><NXS02031><CsaData>4353410C000500C80100000039020000015</CsaData></NXS02031></NXS02XXX>
58	Over Speed Maneuver Statistics		<Module><ID>58</ID><Data><MMOverSpeedManeuverStatistics><TripID>0</TripID><ManeuverID>0</ManeuverID><ManeuverType>Speeding</ManeuverType><StartLocation>0</StartLocation><EndLocation>0</EndLocation><StartTime>0</StartTime><ManeuverDuration>2</ManeuverDuration><SpeedAverage>0</SpeedAverage><SpeedMax>0</SpeedMax><DeltaSpeed>0</DeltaSpeed><TotalManeuverSeverity>Yellow</TotalManeuverSeverity><AverageSpeedInZoneGreen>0</AverageSpeedInZoneGreen><TimeInOverSpeedingZoneGreen>0</TimeInOverSpeedingZoneGreen><TotalEventsIOverSpeedingZoneGreen>0</TotalEventsIOverSpeedingZoneGreen><AverageSpeedInZoneYellow>0</AverageSpeedInZoneYellow><TimeInOverSpeedingZoneYellow>0</TimeInOverSpeedingZoneYellow><TotalEventsIOverSpeedingZoneYellow>0</TotalEventsIOverSpeedingZoneYellow><AverageSpeedInZoneRed>1</AverageSpeedInZoneRed><TimeInOverSpeedingZoneRed>0</TimeInOverSpeedingZoneRed><TotalEventsIOverSpeedingZoneRed>0</TotalEventsIOverSpeedingZoneRed><MaxX>0</MaxX><MaxY>0</MaxY><MaxZ>0</MaxZ><MaxRPM>0</M

Id	Name	XML tag + CellocatorHub parameters	XML
			axRPM><AverageRPM>0</AverageRPM><RiskScore>0</RiskScore></MMOverSpeedManeuverStatistics></Data></Module>
59	RPM Continuous Maneuver Statistics		<Module><ID>59</ID><Data><MMRPMContinuousManeuverStatistics><TripID>0</TripID><ManeuverID>0</ManeuverID><ManeuverType>Excessive_RPM</ManeuverType><StartLocation>0</StartLocation><EndLocation>0</EndLocation><StartTime>0</StartTime><ManeuverDuration>0</ManeuverDuration><MaxRPM>0</MaxRPM><AverageRpmDuringSession>0</AverageRpmDuringSession><AverageSpeedDuringRpmEvent>0</AverageSpeedDuringRpmEvent><MaxSpeedDuringRpmEvent>0</MaxSpeedDuringRpmEvent><TimeInLowRpmCoasting>0</TimeInLowRpmCoasting><AverageSpeedInLowRpmCoasting>0</AverageSpeedInLowRpmCoasting><NumberOfLowRpmCoastingEvents>0</NumberOfLowRpmCoastingEvents><TimeInHighRpmZone>0</TimeInHighRpmZone><AverageSpeedInHighRpmZone>0</AverageSpeedInHighRpmZone><NumberOfHighRpmEvents>0</NumberOfHighRpmEvents><TimeInExcessiveRpmZone>0</TimeInExcessiveRpmZone><AverageSpeedInExcessiveRpmZone>0</AverageSpeedInExcessiveRpmZone><NumberOfExcessiveRpmEvents>0</NumberOfExcessiveRpmEvents><MaxX>0</MaxX><MaxY>0</MaxY><MaxZ>0</MaxZ></MMRPMContinuousManeuverStatistics></Data></Module>
60	Idling Continuous Maneuver Statistics		<Module><ID>60</ID><Data><MMIdlingContinuousManeuverStatistics><TripID>0</TripID><ManeuverID>0</ManeuverID><ManeuverType>Idling</ManeuverType><StartLocation>0</StartLocation><EndLocation>0</EndLocation><StartTime>0</StartTime><ManeuverDuration>0</ManeuverDuration><ShortIdlingDuration>0</ShortIdlingDuration><LongIdlingDuration>0</LongIdlingDuration><SecondsIdlingDetectionBasedGPS>0</SecondsIdlingDetectionBasedGPS><SecondsIdlingDetectionBasedAcc>0</SecondsIdlingDetectionBasedAcc><SecondsEngineIsOnDuringIdlingSession>0</SecondsEngineIsOnDuringIdlingSession><SecondsIgnitionIsOnDuringIdlingSession>0</SecondsIgnitionIsOnDuringIdlingSession></MMIdlingContinuousManeuverStatistics></Data></Module>
61	Off Road Continuous Maneuver Statistics		<Module><ID>61</ID><Data><MMOffRoadContinuousManeuverStatistics><TripID>0</TripID><ManeuverID>0</ManeuverID><ManeuverType>Off_road</ManeuverType><StartLocation>0</StartLocation><EndLocation>0</EndLocation><StartTime>0</StartTime><ManeuverDuration>0</ManeuverDuration><AverageSpeedDuringRpmEvent>0</AverageSpeedDuringRpmEvent><MaxSpeedDuringRpmEvent>0</MaxSpeedDuringRpmEvent><MaxX>0</MaxX><MaxY>0</MaxY><MaxZ>0</MaxZ></MMOffRoadContinuousManeuverStatistics></Data></Module>

Id	Name	XML tag + CellocatorHub parameters	XML
63	Crash Maneuver Statistics		<Module><ID>63</ID><Data><MMCrashManeuverStatistics><TripID>0</TripID><ManeuverID>0</ManeuverID><ManeuverType>Harsh_Brake</ManeuverType><StartLocation>0</StartLocation><EndLocation>0</EndLocation><StartTime>0</StartTime><ManeuverDuration>0</ManeuverDuration><CrashID>0</CrashID><VehicleType>Private</VehicleType><CrashType>Spare</CrashType><MaxG>0</MaxG><CrashInfo>INFO_INIT</CrashInfo><CrashRoll>Roll_Event</CrashRoll><CrashParking>Crash_While_Driving</CrashParking><DurationInSeconds>0</DurationInSeconds></MMCrashManeuverStatistics></Data></Module>

3.4 How to send Commands from the Server to the Unit

As with receiving messages from the unit to the database (as described on page 56), the interface to the Integration Tool commands is the database; commands are managed using your preferred SQL management tool (such as SQL Server Management Studio Tool or MySQL Workbench).

Note that there are two actual methods of sending commands:

- The new command templates introduced in version 2.7.100 (the new command format uses the Commands table and spGenerateCommand store procedure and is available for both the new CellocatorHub database and legacy TWPQueues database). These templates enable users unfamiliar with Cellocator protocols to create commands quickly and easily in the CMUDLQueue table.
- The legacy method (used by **TWPQueues** under the same *Databases* folder and explained on page 57).

3.4.1 *Sending Commands using the Predefined Templates*

The following command templates are included by default with the Integration Tool. See the example following the table for details on how to use a template.

Command	Parameter	Comments
Set_APN	APN string	Generates two programming commands.
Set_Main_Server_IP	Server IP	Server IP should replace dots and enter eight characters.
Set_DNS	DNS	Generates three programming commands.
Set_Port	Port	Four characters.
Set_TCP		
Set_UDP		
Set_Safety_Server_IP	Server IP	Server IP should replace dots and enter eight characters.
Set_ Safety_Port	Port	Four characters.
Set_Safety_TCP		
Set_Safety_UDP		
Set_CPlus_APN	APN string	Generates two programming commands.

Command	Parameter	Comments
Activate_CPlus_Every	Days	
Enable_CPlus_server		Enables all C+ flags at address 1398.
Disable_CPlus_server		Disables all C+ flags at address 1398.
Connect_To_CPlus		Goes to C+ command.
Activate_Led		
Deactivate_Led		
Activate_Immobilizer		
Deactivate_Immobilizer		
Activate_Siren		
Deactivate_Siren		
Activate_Gradual		
Deactivate_Gradual		
Activate_Blinker		
Deactivate_Blinker		
Activate_CFE	CFE IO	1-6 (the I/O index; the CFE supports 6 I/O connections)
Deactivate_CFE	CFE IO	1-6 (the I/O index; the CFE supports 6 I/O connections)
Activate_For_Time_Led	Seconds	
Deactivate_For_Time_Led	Seconds	
Activate_For_Time_Immobilizer	Seconds	
Deactivate_For_Time_Immobilizer	Seconds	
Activate_For_Time_Siren	Seconds	
Deactivate_For_Time_Siren	Seconds	
Activate_For_Time_Gradual	Seconds	
Deactivate_For_Time_Gradual	Seconds	
Activate_For_Time_Blinker	Seconds	
Deactivate_For_Time_Blinker	Seconds	

Command	Parameter	Comments
Activate_For_Time_CFE	CFE IO, Seconds	Handles two parameters.
Deactivate_For_Time_CFE	CFE IO, Seconds	Handles two parameters.
ITMobile_CAR_Sharing_Lock		
ITMobile_CAR_Sharing_UnLock		
Status_Request		
Status_Request_Safety		
Device_Reset		
GPS_Reset		
Erase_Log_From_Unit		
Erase_Safety_Log_From_Unit		
Request_Cell_ID		
Transparent_Mode_Start		
Transparent_Mode_Stop		
Request_accident_raw_upload		
Safety_Erase_unsent_CSA_events		
Safety_Delete_all_files_other_then_EDR_files		
Safety_Erase accident data		
Request_Calibration_Status		
Leave_calibration_mode		
Enter_calibration_mode		
Enter_offroad_ready_mode		
Restore_Preset_Command_Private		
Restore_Preset_Command_Large_Van		
Restore_Preset_Command_Light_Truck_Bus		
Restore_Preset_Command_Heavy_Truck		

3.4.1.1 Example of a predefined template

The following example shows how to use a template (highlighted in red):

1. Display the commands table using the following script:

```
SELECT * FROM [CellocatorHub].[dbo].[Commands]
```

2. Select from the list the relevant command; note that some have parameters and to learn more about each command parameter read the command comment and parameter description columns.
3. Execute the `spGenerateCommand` with the relevant parameters (as described in the *Command Message Interface* table below). The execution of the `spGenerateCommand` generates a downlink message in the DownlinkQueue table that will be processed and sent to the unit.

   The following example shows how to set the main server IP to 201.235.101.72:

```
USE [CellocatorHub]
GO

DECLARE       @return_value int

EXEC   @return_value = [dbo].[spGenerateCommand]
               @CMUId = 9000,
               @Command = N'Set_Main_Server_IP',
               @Param = N'201.235.101.72',
               @LinkerAppId = 1,
               @CorrelatorAppId = 1,
               @MsgSN = 100,
               @NetworkType = 4

SELECT 'Return Value' = @return_value
```

3.4.1.2 Command Message Interface

Field Name	Field Description	Type	Example
Command	The command name that can be obtained from the "Commands" table, "Command" parameter.	String	@Command = N'Activate_Blinker'
Param	Parameters for the command; can be null, single or multiple comma separated parameters, according to the description in the "Comments" column of the "Commands" table.	String	??
CMUId	Cellocator unit's ID as it appears on unit exterior.	Integer	152321
SrvrId	Should include the Linker Instance ID, if there is more than one Linker, otherwise NULL.	Integer	NULL

Field Name	Field Description	Type	Example
CorrelatorAppId	Instance Id of the Correlator, corresponds with the CorrelatorAppID in the Correlator INI file.	Integer	0
MsgSN	The serial number of the command. When the unit sends a reply to a specific command, it includes this SN in the message. This way it is possible to match the Command and the Ack Message sent by the unit.	Integer	0-255
NetworkType	The preferred medium to send command [4-GPRS, 5-SMS].	Integer	4
PhoneCountry		String	NULL
PhoneRegion		String	NULL
SMSPhoneNum		String	NULL
UnitIP		String	NULL
CallbackPhoneNo		String	NULL

3.4.2 *Sending Commands using the Legacy TWPQueues*

3.4.2.1 Downlink Message Interface

All the fields in the following table are mandatory for the CorrelatorMax application:

Field Name	Field Description	Type	Example
LineNumber	Index	Long	
SystemId	Unit ID in the client system (can be set as the same as the CMUId).	Integer	152321
CMUId	Cellocator unit's ID as it appears on unit exterior.	Integer	152321
MsgSN	The serial number of the command. When the unit sends a reply to a specific command, it includes this SN in the message. This way it is possible to match the Command and the Ack Message sent by the unit.	Integer	0-255

Field Name	Field Description	Type	Example
MsgType	Command type according to the protocol (see the *Integration Package OTA Compatibility* section for the list of types supported – MCGP or CSA). See also the *Custom Command* section in the following *Examples of Integration Commands*.	Integer	0-255 or 256-511
MsgBody	Command body without header, authentication code, repetitions, and error detection code - see protocol MCGP or CSA. See also the *Custom Command* section in the following *Examples of Integration Commands*.	Integer	020100003C0000 – Emergency command 1 Tx/Min
MsgACKRequest	Always 8	Integer	8
ConfirmNum	Future use=> always 1	Integer	1
RMUType	Always 60 - Cellocator	Integer	60
NetworkType	The preferred medium to send command [4-GPRS, 5-SMS].	Integer	4
DateOfEntry	Command insert time	Date Time	'2009-06-21 11:20:15.720 '
PhoneCountry		String	NULL
PhoneRegion		String	NULL
SMSPhoneNum		String	NULL
UnitIP		String	NULL
CallbackPhoneNo		String	NULL
SrvrId	Should include the Linker Instance ID, if there is more than one Linker, otherwise NULL.	Integer	NULL
ProtocolType	Wireless protocol type: MCGP or CSA	Small integer [2 bytes]	default
CorrelatorAppId	Instance Id of the Correlator, corresponds with the CorrelatorAppID in the Correlator INI file.	Integer	0

3.4.2.2 Examples of Integration Commands

This section includes examples of the following integration commands:

- **Custom Command** see below.
- **MCGP/S Generic Command - Message Type 0**, page 115
- **MCGP/S Programming Command - Message Type 1**, page 117
- **MCGP/S 'Forward Data' Message - Message Type 5**, page 120
- **MCGP/S Modular Command - Message Type 9**, page 122
- **MCGP/S New Modular Command - Message Type 11**, page 124
- **CSA Protocol Frame**, page 129

Custom Command

For more generic formats a Msg type of 256-511 can be used according to the following description: if `MsgType` bit 8 is 1, a message will be created based on the structure in the following table.

1	System Code, byte 1 – ASCII "M"	Constant
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 (X)	`MsgType` & 0xFF
6	Destination Unit's ID (total 32 bits)	`CMUId`
7		
8		
9		
10	Command Numerator	`MsgSN`
11	Authentication	0, recalculated at Linker level.
12		
13		
14		
15	Data	`MsgBody`
..		
	Checksum	Auto Calculated

The CorrelatorMax also supports "Custom" commands.

- MsgType should be sent as 0x100 + Msg type (for example, msg type 0 -> 0x100 ->256, msg type 9 -> 0x109 -> 265, msg type 11 - > 0x10B -> 11 -> 267)

- The payload is all the data besides the 14 bytes header (MCGP+Type+UnitID+Numerator+AuthCode) and without the checksum (for example in msg type 9 it will include the packetcontrol byte, total length, module id, module length and module payload).
- SQL Example:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum]
,[RMUType],[NetworkType],[DateOfEntry], [ProtocolType])
VALUES
(50999970,9999,00,265,'800D1D0B128B3E022E000000000000',8,1,60,4,getdate(),0)
```

MCGP/S Generic Command - Message Type 0

The Generic command has a predefined length of 25 bytes. It contains the following data (listed in the actual transmitted order):

1. Message header
   - System code – 4 bytes
   - Message type – 1 byte
   - Target Unit's ID – 4 bytes
   - Command Numerator Field
2. Authentication code – 4 bytes
3. Command data:
   - Command code field – 1 byte, repeats twice.
   - 1st Command data field – 1 byte, repeats twice.
   - 2nd Command data field – 1 byte, repeats twice.
   - Command Specific Data - 4 bytes
4. Error detection code – 8-bit additive checksum (excluding system code)

As stated in the *Downlink Message Interface* section, the MsgBody field should include the command body without header, Authentication code, repetitions, and error detection code; for programming the command, we need to take only part 3 of the message: Command data.

The Command data section includes the following (7 bytes total - repetitions are excluded):

- Command Code – since the generic command includes different kinds of commands, this byte holds a unique command code which is used to specify the command to be executed. A list of command codes can be found in the Cellocator OTA protocol.
- Command Data (1st and 2nd) - the command data fields contain further information which is needed by some of the commands.
- Command Specific Data - the command data field (4 bytes) contains additional information, specified separately for each command code.

Example of 'Emergency by Command'

According to the OTA protocol, the command code = **0x02**, the First command data = **0x01**, the Second command data is not in use, so= **0x00**, and the Specific Data includes the following parameters:

Number of distress trans.=**0** (infinite), time between distress trans. events=**5sec**; the two other bytes are not in use, so the value is = **0x00050000**

Concatenating the above will result with: **02010000050000**

Example for 'Status Request'

According to the OTA protocol, the command code = **0x00**, and the rest are not in use, so the values is 6 zero bytes: **0x000000000000**.

Concatenating the above will result with: **00000000000000**

Example for 'Reset Command'

According to the OTA protocol, the command code = **0x02**, the First command data = **0x02**, and the rest are not in use, so the values is 5 zero bytes: **0x0000000000**.

Concatenating the above will result with: **02020000000000**

To insert a Generic Command into the CMUDLQueue table run the following statement.

To insert a GPRS reset command run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])
VALUES
(300000,300000,10,0,'02020000000000',8,1,60,4,getdate(),972,54,000888,0)
```

To insert a SMS reset command run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])
VALUES
(300000,300000,10,0,'02020000000000',8,1,60,5,getdate(),972,54,000888,0)
```

MCGP/S Programming Command - Message Type 1

The programming command has a predefined length of 34 bytes. It contains the following data (listed in the actual transmitted order):

1. Message header
   - System code – 4 bytes
   - Message type – 1 byte
   - Target Unit's ID – 4 bytes
   - Command Numerator Field
2. Authentication code – 4 bytes
3. Memory data
   - Block code – 1 byte
   - Programming "masking" bitmap – 2 bytes
   - Block data – 16 bytes
4. Error detection code – 8-bit additive checksum (excluding system code)

As stated in the *Downlink Message Interface* section, the MsgBody field should include the command body without header, Authentication code, repetitions and error detection code; so, for programming the command we need to take only part 3 of the message: **Memory data**.

The Memory data section includes the following (19 bytes in total):

- **Block code** – Programming commands basically rewrite the unit's EEPROM. When done OTA, EEPROM access is performed in blocks. The 2031 bytes EEPROM space is partitioned to blocks of 16 bytes. This means the whole EEPROM space contains 176 different blocks, assigned with block codes of 0 (zero) to 175 (decimal). The first block (which represents EEPROM locations 0 to 15 decimal) is assigned with block code 0 (zero). The following blocks are assigned with successive numbers (block 1 for locations 16 to 31 and so on).
- **Programming "masking" bitmap** - The bitmap allows programming only part of the parameters in a block, while leaving the other parameters with their previous value. The 2 bytes of the masking bitmap are mapping the block being accessed. Each bit in the 16-bit (2-bytes) value represents a byte in the parameter's memory block. The LS bit of the bitmap represents the byte with the lowest offset in the program block. A value of "1" in a certain bit enables programming to the corresponding byte in the parameter's memory, whereas a value of "0" prohibits programming of that byte. Note that since the programming is done in bytes, if you need to update a specific bit you MUST make sure that the rest of the byte includes the original value from the unit in order to keep the other parameters intact.
- **Block data** – Contains the actual data programmed in the specified block of the parameter memory.

Garmin parameter is 1 bit and located at address D'1348' bit address 2 - from the Cellocator Programmer/Programming Manual.

Block code: 1348\16 → 84.25 → 84 is block 0x54 and .25*16 is byte 4.

Masking: Need to update only byte 4 in the block of 16-bytes

00001000 00000000 → mirroring the mask 00010000 00000000 → 10 00

Block Data: Since the masking includes only byte 4, the rest of the block is meaningless; you can leave all other bytes as zero or any other value. They will be ignored.

```
Disable Garmin→  00000000 (bit 2 set to 0) → 00
Enable Garmin→ 00000100 (bit 2 set to 1)→ 04
```

Again, note that you should know first the actual value of the masked byte and then modify the proper bit only, the rest should stay as is.

So, in order to enable the Garmin, concatenating the above will result with:

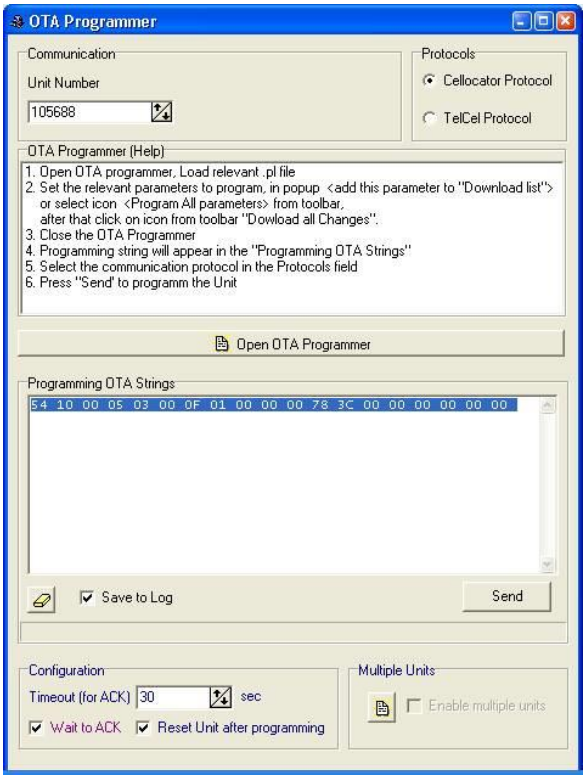
5410000000000004000000000000000000000000

To disable the Garmin, concatenating the above will result with:
5410000000000000000000000000000000000000

Example for programming block using the Communication Center:

Note that each programming command can be generated while using the Communication Center; this will make the integration easier, although you are still required to use the exact PL configured in the units when building the commands. Otherwise, parameters in the same programming block might get overridden with the values included in the Communication Center.

The following dialog box includes the programming block, created by the OTA Programmer in the Communication Center. You can copy the OTA string, trim the spaces and include it in the programming command.



NOTE: After sending the programming command you need to wait for an Ack. (Msg. Type 3) from the unit with the same SN number and then send a Reset command so the unit will load the new values from the EEPROM.

To insert a Programming Command into the CMUDLQueue table run the following statement.

To insert a GPRS programming command run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]

([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])

VALUES

(300000,300000,15,1,'5410000000000004000000000000000000000000',8,1,60,4,getdate(),972,54,000888,0)
```

After sending this command, wait for a GRPS Uplink, Message Type 3, with the same SN (15) and then send a Reset Command; see Message Type 0 above for specific instructions.

To insert a SMS programming command run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]

([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])

VALUES

(300000,300000,15,1,'5410000000000004000000000000000000000000',8,1,60,5,getdate(),972,54,000888,0)
```

After sending this command, wait for a SMS Uplink, Message Type 3, with the same SN (15) and then send a Reset Command; see Message Type 0 above for specific instructions.

MCGP/S 'Forward Data' message - Message Type 5

The forward data command has a varying length up to 217 bytes. It contains the following data (listed in the actual transmitted order):

1. Message header
   - System code – 4 bytes
   - Message type – 1 byte
   - Target Unit's ID – 4 bytes
   - Command numerator – 1 byte
2. Authentication code – 4 bytes
3. Setting Byte – 1 byte
4. Data length – 1 byte
5. Data to Forward – variable up to 199 bytes
6. Error detection code – 8-bit additive checksum (excluding system code)

As stated in the *Downlink Message Interface* section), the MsgBody field should include the command body without header, Authentication code, repetitions, and error detection code; so, for data forward messages we need to take parts 3 to 5 of the message: **Setting Byte**, **Data length** and **Data to Forward**.

Setting Byte - This byte is used for different system indications: Packet to NavMan (MDT) should be set to "0", otherwise, set to "1" if the packet should be forwarded to a Garmin terminal or any other device.

Data Length – Calculated by the CorrelatorMax and should be omitted from the message string.

Data to Forward – This is the data that is forwarded to the terminal attached to the unit. This field must be an exact number of bytes long, as listed in the Data Length field. Note that the format of this data should be according to the specific protocol (i.e., MDT, Garmin etc.).

Example of Forward Data Message to NavMan:

Setting Byte: Set to **0x00**, as it is not Garmin.

Data Length: According to the length of the data to forward; see below.

Data to Forward: In the MDT protocol the Data to Forward section includes the following:

Packet Data

- MCB (Message Control Byte) - 1 byte
- Message Numerator - 1 byte
- Message 198 bytes (max.)

Where MCB is:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved for future use				NAK	ACK	Always sent logic one (1) from the central server.	Message box 1 => Inbox (Regular Text Message or a Question). 0 => Preset box (Answer).

Message Numerator – a sequential number of the MDT message sent; a value between 0 and 255.

Message – is the actual text message, each character represented by an ASCII code, converted to HEX.

When sending a preset (answer) message the format of the message box is composed of the following parts:

- Preset Message Header - 8 bytes
- Preset Message Delimiter (0x20) - 1 byte
- Preset Message Body – 50 bytes (max)

NOTE: ACK\NAK messages are handled automatically by the Linker; always set ACK & NAK to 0, for regular text messages.

In order to send a question with a few possible answers, first send the question with byte0 set as 1, and then send additional messages for each possible answer with this byte set to 0.

Example for regular MDT text message:

The text to be sent is: Please Call Me!

Converting the text to hex:

ASCII	P	l	e	a	s	e		C	a	l	l		M	e	!
HEX	50	6C	65	61	73	65	20	43	61	6C	6C	20	4D	65	21

MCB byte: 00000011 -> **0x03**

Concatenating the above will result with:

00**03**01506C656173652043616C6C204D6521

Example for possible Answer in MDT message:

The text to be sent is Yes in the message body and in the message header.

Converting the text to hex:

ASCII	Y	e	s							Y	e	s
HEX	59	65	73	20	20	20	20	20	20	59	65	73

MCB byte: 00000010 -> **0x02**

Concatenating the above will result with:

00**02**025965732020202020596573

To insert a Forward Data Message into the CMUDLQueue table run the following statement.

To insert a GPRS Forward Data Message, run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum],[ProtocolType])
VALUES
(300000,300000,18,5,'000301506C656173652043616C6C204D6521',8,1,60,4,getdate(),972,54,000888,0)
```

To insert a SMS Forward Data Message, run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])
VALUES
(300000,300000,18,5,'000301506C656173652043616C6C204D6521',8,1,60,5,getdate(),972,54,000888,0)
```

MCGP/S Modular Command - Message Type 9

The modular data packet request is designed to provide different data types in the same packet. The modular data request contains the following bytes (listed in the actual transmitted order):

1. Message header
   - System code – 4 bytes
   - Message type – 1 byte
   - Target Unit's ID – 4 bytes
   - Command Numerator Field – 1byte
   - Spare – 4 bytes
2. Packet Control Field – 1 byte
3. Total Length – 1 byte
4. First Sub-Data Type– 1 byte

   First Sub-Data Length – 1 byte

First Sub-Data– variable length, depends on Data Type.

……

5. Nth Sub-Data Type – 1 byte (option)

   Nth Sub-Data Length – 1 byte

   Nth Sub-Data– variable length, depends on Data Type N

6. Error Detection Code – 8 bit additive

The MsgBody field should include the command body as explained below **without** header, Total Length, Sub-data length, Authentication code, repetitions, and error detection code; for modular commands we need to take only **Packet Control Field, Sub-Data Type** & **Sub Data data**.

Packet Control Field - The Packet Control Field should always be **82** for requests sent to the unit.

Sub-Data – This is the relevant Sub-data module that is requested in this command. It should include the sub-data type and actual data, as defined in the protocol. A value of 30 (Hex) should be added to the Sub-Data Type field.

Example for 'Modular Platform Manifest (sub-data 0x12)'

According to the OTA protocol this command causes the unit to generate an OTA Modular Platform Manifest message. The message will contain the data fields as per the specification in a command.

Data part: The data part of this packet has a size of 6 bytes. Each byte contains a bitmask as described below. Setting the bit to "1" causes the unit to add a corresponding field to the Modular Platform Manifest.

Assuming we would like the full bitmask, we should send it as follows:

Type=12 (A value of 30 (Hex) should be add which results in 42 (Hex).

Data= FFFFFF (for full bitmask) and 000000 (for the last 3 bytes)

Concatenating the above will result with: **42FFFFFF000000**

To insert a Modular Command into the CMUDLQueue table run the following statement.

To insert a GPRS reset command, run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])
VALUES
(300000,300000,10,9,'8242FFFFFF000000',8,1,60,4,getdate(),972,54,000888,0)
```

To insert a SMS reset command, run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
```

```
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])

VALUES

(300000,300000,10,9,'8242FFFFFF000000',8,1,60,5,getdate(),972,54,000888,0)
```

MCGP/S New Modular Command - Message Type 11

Type 11 was introduced to support true modular protocols. The basic structure of the protocol is designed to carry records with predefined structure called modules. The protocol will be used as an extension for the CelloTrack protocol. Type 11 supports theoretical message lengths of up to 65536 bytes, though the actual rate will be constrained by the hardware limitations. It contains the following data (listed in the actual transmitted order):

1. Message header
   - System code – 4 bytes
   - Message type – 1 byte
   - Target Unit's ID – 4 bytes
   - Command Numerator Field – 1byte
2. Authentication Code – 4 bytes
3. Packet Control Field – 1 byte
4. Total Length – 2 bytes
5. Spare – 4 bytes
6. First Module

   ……
7. Nth Module
8. Error Detection Code – 8 bit additive

As stated in the *Downlink Message Interface* section, the MsgBody field should include the command body **without** header, Authentication code, repetitions, and error detection code; so, for modular commands we need to take only parts 3 to 7 of the message: **Packet Control Field**, **Total length**, **Spare** and **Modules**.

Packet Control Field - The Packet Control Field should always be **80** for requests sent to the unit.

Total Length – should include the total modules length + the 4 spare bytes.

Spare – 4 bytes, set as zeros.

Modules– This is the relevant Sub-data module that is requested in this command. It should include the sub-data type, length, and actual data, as defined in the protocol.

Example for 'General Module Query' request (Module ID 29)

According to the OTA protocol this command will be sent by the server to request a set of modules to be returned to the server. The module describes a list of module Ids. The message will contain the data fields as per the specification in a command; **Module ID, Length, Number of Requested Module, requested Module Ids.**

Assuming we would like the full bitmask, we should send it as follows:

Module ID=1D (29)

Length=03 (in two bytes)

Number of Requested Modules= 2

Requested Module Ids = 04, 06

Concatenating the above will result with: **1D030002041F**

Combining the above the **MsgBody** with the Module ID 29 request should look like:

800A00000000001D0300020406

PacketControlField,TotalLength (6 for module+4 spare),Spare,Module29

In order to insert a Modular Command into the CMUDLQueue table run the following statement.

To insert a GPRS reset command run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])
VALUES
(300000,300000,10,11,'800A00000000001D0300020406',8,1,60,4,getdate(),972,54,000888,0)
```

To insert a SMS reset command run:

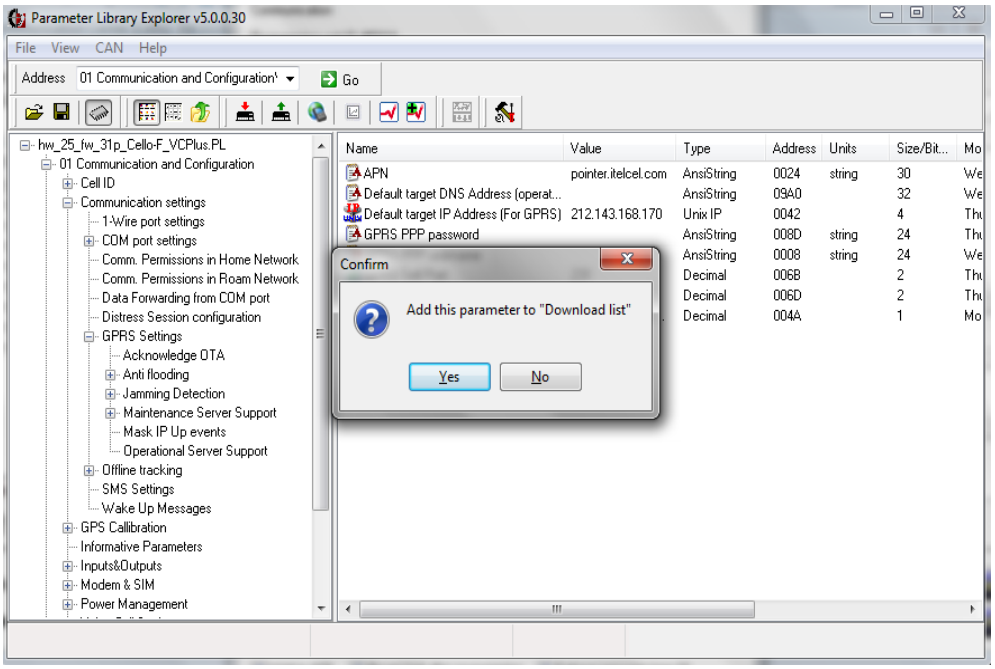
```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])
VALUES
(300000,300000,10,11,'800A00000000001D0300020406',8,1,60,5,getdate(),972,54,000888,0)
```

Example for 'Configuration Memory Write Module' (Module ID 10) using the Communication Center.

Note that each programming command can be generated while using the Communication Center (part of the Cellocator Evaluation Suite).

When opening the OTA programmer dialog, make sure to select the **Enforce message Type 11** checkbox at the bottom of the screen, so the programming commands will be created in the new format.

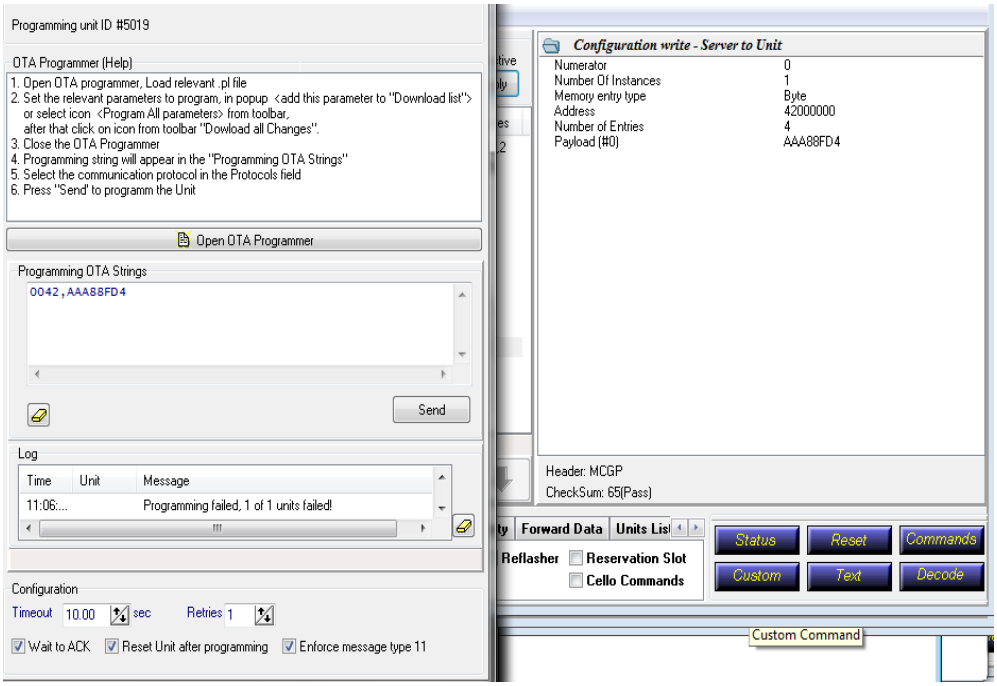
In the OTA programmer dialog change the relevant parameters and click Yes to add the parameter to the download list. In the example below the Operational Server IP was changed:



When finished, click 'Download all changes' to get the programming strings.

The following dialog box includes the programming strings, created by the OTA Programmer in the Communication Center. The string is composed of the 'Initial address' and the actual buffer (data to be updated).

Clicking the Send button will allow you to see the command parameters required for this command, as shown below.



According to the OTA protocol this command will update the buffer data in the specified address location.

The module requires the following data: **Module ID, Length, Numerator, Number of Instances, Memory Type, Memory entry Unit Type, Address, Number of Entries, Buffer.**

Assuming we would like the programming string created above we should send it as follows:

Module ID=0A (10)

Length=15 (in two bytes)

Numerator= 0 (in two bytes)

Number of Instances = 1

Memory Type = 0 (always)

Memory Entry Unit Type = 1 (for Byte)

Address=42000000

Number of Entries = 04 (in two bytes)

Buffer=AAA88FD4

Concatenating the above will result with: **0A0F0000000100014200000000400AAA88FD4**

Combining the above the **MsgBody** with the Module ID 10 request should look like:

8016000000000000A0F0000000100014200000000400AAA88FD4

PacketControlField,TotalLength (18 for module+4 spare),Spare,Module10

In order to insert a Modular Command into the CMUDLQueue table run the following statement.

To insert a GPRS reset command, run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]

([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmN
um],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[S
MSPhoneNum], [ProtocolType])

VALUES

(300000,300000,10,11,'80160000000000A0F0000000100014200000000400AAA88FD4'
,8,1,60,4,getdate(),972,54,000888,0)
```

To insert a SMS reset command, run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]

([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmN
um],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[S
MSPhoneNum], [ProtocolType])

VALUES

(300000,300000,10,11,'80160000000000A0F0000000100014200000000400AAA88FD4’
,8,1,60,5,getdate(),972,54,000888,0)
```

Example for 'General Command' request (Module ID 32)

According to the OTA protocol this command will be sent by the server to request a set of modules to be returned to the server. The module describes a list of module Ids. The message will contain the data fields as per the specification in a command, **Module ID, Length, Number of Command Entries, command Entries.**

We should send it as follows:

Module ID=20 (32)

Length=05 (in two bytes)

Number of Requested commands= 1

Requested commands Entries (Command ID (2 bytes), Command Data Bytes (if relevant)) = 0100 (no data bytes in reset commands)

Concatenating the above will result with: **200300010100**

Combining the above the **MsgBody** with the Module ID 32 request should look like:

800A00000000000200300010100

PacketControlField,TotalLength (6 for module+4 spare),Spare,Module29

To insert a Modular Command into the CMUDLQueue table run the following statement.

To insert a GPRS reset command, run:

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]

([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])

VALUES

(300000,300000,10,11,'800A00000000000200300010100',8,1,60,4,getdate(),972,54,000888,0)
```

CSA Protocol Frame

The CSA protocol frame contains the following data (listed in the actual transmitted order):

<table>
<tr><td>1</td><td>System code, byte 1 – ASCII “C”</td><td rowspan="13">Prefix of message's frame</td></tr>
<tr><td>2</td><td>System code, byte 2 – ASCII “S”</td></tr>
<tr><td>3</td><td>System code, byte 3 – ASCII “A”</td></tr>
<tr><td>4</td><td rowspan="2">Length of message from byte 6 to CS</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td><td rowspan="2">Message ID (sequential numerator used by ACK mechanism. Replies contain the numerator of the command) - SN</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td>Message Type / Initiator / Direction (0 for CSA event)
<table>
<tr><td>Initiator
0 – reply or ack
1 - active</td><td>Direction
0 – inbound
1 - outbound</td><td colspan="6">Message Type</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
</table></td></tr>
<tr><td>9</td><td rowspan="4">Cello Unit’s ID (total 32 bits)</td></tr>
<tr><td>10</td></tr>
<tr><td>11</td></tr>
<tr><td>12</td></tr>
<tr><td>13-…</td><td>First Module</td><td rowspan="3">Payload of the message</td></tr>
<tr><td>…</td><td>Optionally second module</td></tr>
<tr><td>…</td><td>….</td></tr>
<tr><td>…</td><td>Error detection code – 8-bit additive checksum</td><td>Suffix of message's frame</td></tr>
</table>

As stated in the *Downlink Message Interface* section, the MsgBody field should include the Module itself (Module ID, Module Length, No. of request Module, Requested Module).

According to the CSA OTA protocol:

- System Code (3 bytes) – added automatically by the GW applications.
- Length of Message (2 bytes) - added automatically by the GW applications.
- MessageID - should be added to ‘MsgSN’ field.
- Message Type - should be added to ‘MsgType’ fields.
- Initiator / Direction - added automatically by the GW server.
- Cello Unit Id – should be added to ‘CMUId’ field.

- ModuleID + Module Length + No. of request Module + Requested Module – should be concatenated and added to 'MsgBody' field.

Example for 'Status Request' for 'CSA FW ID'

13	Module ID - 12	Payload of the message
14	Module Length	
15	Number of request Module	
16	Request Module	

- System Code: "CSA"
- Length of Message: '10' (decimal)
- MessageID: "9"
- Message Type / Initiator / Direction: 'Command to CSA' is '4' (Binary –100) (Initiator/Direction – [binary '10'] – added automatically)
- Cello Unit Id: '1158' (decimal)
- ModuleID: '12'
- Module Length: '2'
- No. of request Module: '1'
- Requested Module '8' (CSA FW ID)

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum],[RMUType],[NetworkType],[DateOfEntry],[PhoneCountry],[PhoneRegion],[SMSPhoneNum], [ProtocolType])

VALUES
(1158,1158,9,4,'0C020108',8,1,60,4,getdate(),null,null,null,1)
```

Example for Program/Read Parameters to CSA (Message Type 2 from Server to CSA)

This is the structure of the '**Programming Frame**':

0	Module's ID (**10 - Programming Frame**)
1	Length of module
2	Programming command numerator
3	

4	• Action byte (Read/Write/Lock/Unlock) • 0 for Read command. • 1 for Write command. • 2 for Lock command (an infrastructure - currently not used) • 3 for Unlock command (an infrastructure - currently not used)
5	The first address
6	
7	
8	Length of data
9	The data (in case of Read programming - single byte of Zero)
10	

- System Code: "CSA"
- Length of Message: '10' (decimal)
- MessageID: "9"
- Message Type / Initiator / Direction: 'Programming inbound' is '2' (Initiator/Direction – [binary '10'] – added automatically)
- Cello Unit Id: '1158' (decimal)
- ModuleID: '10' (hex 0A)
- Module Length: '8'
- Programming command numerator: '15'
- Action Byte: '1' (write command)
- First Address: 104 (hex 68)
- Length of Data: 1
- Data: 3

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]
([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum] ,[RMUType],[NetworkType],[DateOfEntry], [ProtocolType])

VALUES

(1158,1158,9,2,'0A08000F016800000103',8,1,60,4,getdate(),1)
```

Example for 'Restore Default Vehicle Preset' (module 56)

Byte	Description	Default
0	Module Name 56 – Restore vehicle preset	56
1	Length of module – 2	2
2	Preset number. Number: Vehicle type 0: Private 1: Large Van 2: Light Truck/bus 3: Heavy Truck	0
3	Spare	0

- System Code: "CSA"
- Length of Message: '10' (decimal)
- MessageID: "9"
- Message Type / Initiator / Direction: 'Programming inbound' is '2' (Initiator/Direction – [binary '10'] – added automatically)
- Cello Unit Id: '1158' (decimal)
- ModuleID: '56' (hex 38)
- Module Length: '2'
- Preset Number: '03' - Heavy Truck
- Spare: '0'

```
INSERT INTO [TWPQueues].[dbo].[CMUDLQueue]

([SystemId],[CMUId],[MsgSN],[MsgType],[MsgBody],[MsgACKRequest],[ConfirmNum] ,[RMUType],[NetworkType],[DateOfEntry], [ProtocolType])


VALUES

(1158,1158,9,2,'38020300',8,1,60,4,getdate(),1)
```