Cellocator Integration Package DLL Edition Manual

Proprietary and Confidential

Version 1.1

Revised and Updated: January 30, 2014

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

Table of Contents

1 Introduction

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

The Cellocator Gateway DLL file contains an extensive set of functions that enable a quick and easy integration of the Cellocator OTA protocol.

The integration package is built from a GPRS communication manager DLL, which uses Microsoft Windows WinSocket 2.0 to capture incoming UDP or TCP packets. The library is multi-threaded and runs in parallel to the thread (application) that loads it. The DLL communicates with the application via standard Win32 library function calls and Windows Messages for a-synchronic communication.

1.1 Package Contents

The Cellocator Integration Package consists of the following components:

- GPRSManager.dll v2.1.0.46
- kl2dll32.dll v1.0
- CelloCrypt.dll v1.0
- GPRSManagerSetting.ini
- GPRS_Manager_DLL.map
- Testapplication.exe

1.2 Terminology

Term	Description
CCC	Command & Control Center
Uplink	Message which originates from Cellocator unit and is sent towards the GPRS gateway
Downlink	Message which originates at the GPRS gateway and is sent towards Cellocator unit.
MSMQ	Microsoft Message Queuing
GPRS	General Packet Radio Service

1.3 Package Prerequisites

The Cellocator Gateway Integration Package requires the following prerequisites in order to utilize the library:

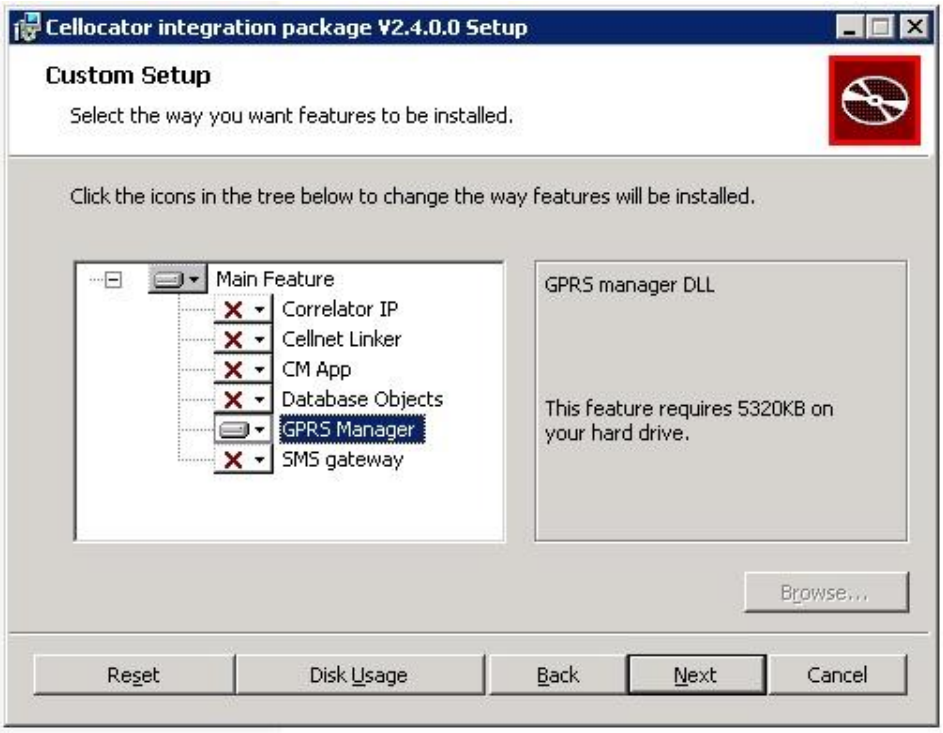
- Win32 compatible operating system (Windows 95 and higher)
- Free communication port
- Valid license file (License.db)

2 Package Installation

The following procedure describes how to install the Cellocator Integration Package.

➢ **To install the Integration Package:**

1. Run the installer package by double-clicking the **Integration package installer.msi**. The setup wizard is launched.
2. In the displayed *Welcome* screen, click **Next**.
3. In the displayed *License Agreement* screen, select the **I accept the terms in the License Agreement** checkbox, and click **Next**.
4. In the displayed *Choose Setup Type* screen, click **Custom** and then click **Next**.
5. In the displayed *Custom Setup* screen, select the GPRS Manager, and then click **Next**.



6. In the displayed *Ready to install Cellocator Integration Package* screen, click **Install**.
7. When the installation has completed, click **Finish**.

   The package is located in C:\Program Files\Cellocator\Integration Package\GPRS Manager.
8. Copy the **license.db** file to the GPRS Manager folder.

3 Communication Mechanism

The library uses an a-synchronic communication engine which is capable of sending and receiving messages at any time. Transmission of messages is done using a standard call to one of the library's relevant functions.

The reception procedure works in a manner that once a message is received by the DLL and passes the error correction test, the library sends a Windows message to the application that informs the application that a new message is waiting for peeking using the relevant functions.

Note that several functions exist to send a message; the difference is in the transmission result information. For more information refer to each function's details in the following sections.

3.1 Functions

This section lists the library functions, including their parameters and return values. Note that several functions refer to constants, which are described in the *Constants* section.

3.1.1 *InitializeLibrary*

This function initializes the library, and creates both the library engine and the UDP or TCP server.

```
BOOL InitializeLibrary (
                HWnd        AppHandle
                LPCTSTR     Port
                DWORD Flags
                        );
```

Parameters

AppHandle

A handle to an application window. This window should receive the Windows messages indicating that a new message is available, etc.

Port

A pointer to a null terminated string with the port to listen to.

Flags

This can be a combination of the following values:

VERIFY_HEADER - If marked the library will check the header validity for all incoming messages.
VERIFY_CHECKSUM - If marked the library will check the checksum validity for all incoming messages.
LOAD_UNIT_LIST_FROM_FILE - Continues a previous session.
SAVE_UNIT_LIST_TO_FILE - Keeps records of the current session for future use or quick start.
NO_UDP_SERVER - No UDP server is to be created; the library will be able to send only. Use this if you know that a UDP server is already running for the required port.
REPLY_TO_INTEROGATOR_ONLY - If the server is also used as a software NAT, when a unit replies to a command only the unit that issued the command will receive the reply and not the application that runs the library.

ALLOW_SERVICE_TO_ALL_CLIENTS - If the server is also used as software NAT, only the allowed address will receive service.

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call the Windows *GetLastError* function to get the error code.

Remarks

This function must be called before any attempt to use the library. The function will fail if any unexpected error occurs or the UDP server is already in use. If the function fails, there is no need to call the *TerminateLibrary* function (see below).

3.1.2 *TerminateLibrary*

This function terminates the library, clearing the UDP server and releasing the memory used by the library.

```
BOOL TerminateLibrary(void);
```

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call the Windows *GetLastError* function to get the error code.

Remarks

This function should be called when the application decides to stop using the library. When the application is closed the function will be automatically called. Note that all the messages waiting in the incoming messages buffer will be erased.

3.1.3 *SendUnitMessage*

This function sends a message to a known (already handled) unit and will not return until either the unit replied or the no reply timeout of 5 seconds elapsed.

```
BOOL SendUnitMessage (
                LPCTSTR     Msg
                INT         MsgLength
                UINT        UnitNum
                          );
```

Parameters

Msg
: Pointer to the unit message.

MsgLength
: Length of the Unit Message in bytes including the sync code and the checksum.

UnitNum
: The end unit number.

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call the Windows *GetLastError* function to get the error code. If the specified unit is not in the known units list or the unit did not respond the function will fail.

Remarks

Since this function will not return until a transmission result is available the application calling thread will be suspended until then. If the application is not multi-threaded by nature it could mean the entire application will be suspended. For transmission without the wait, use the *PostUnitMessage* function instead (see below).

3.1.4 *PostUnitMessage*

This function will post a message to a known (already handled) unit and will return immediately.

```
BOOL PostUnitMessage  (
                LPCTSTR     Msg
                INT         MsgLength
                UINT        UnitNum
                          );
```

Parameters

Msg
Pointer to the unit message.

MsgLength
Length of the Unit Message in bytes including the sync code and the checksum.

UnitNum
The end unit number.

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call the Windows *GetLastError* function to get the error code. Note that the function will fail if the specified unit is not in the unit list.

Remarks

This function will add a message to the library outgoing messages queue but will not wait for the transmission result. Instead, the transmission result will be sent to the application as the Windows message once it is available. To send a message and to receive the result immediately, use the *SendUnitMessage* function.

3.1.5 *PeekUnitMessage*

This function will pick the first incoming message out of the incoming messages buffer.

```
INT      PeekUnitMessage (
               LPCTSTR     Msg
               INT         MaxLength
                  );
```

Parameters

Msg
Pointer to a buffer.

MaxLength
Size of the buffer. If the message is longer than the size of the buffer no bytes will be copied. The buffer should be equal to the Message length indicated by the GLM_UNIT_MSG_RECEIVED Windows message in the wParam.

Return Value

The function returns the length of the message. If the message length is equal or smaller than the buffer size then the function will succeed, however if the buffer is too small no bytes will be copied and the function will fail.

Remarks

This function should usually be called as a reply to the GLM_UNIT_MSG_RECEIVED Windows message sent by the library. The message will hold the newly received message length in the wParam.

Note that the received messages will be stored in the incoming messages buffer. The buffer size is limited only to the amount of resources of the host machine.

3.1.6 *ForceUnitAddr*

This function will change the address of the specified unit in the library records.

```
BOOL ForceUnitAddr (
                UINT        UnitNum
                LPCTSTR     Address
               );
```

Parameters

UnitNum
Specify the end unit number.

Address
Pointer to a null terminated string specifying the end unit IP address (e.g. 192.168.0.1).

Return Value

If the function succeeds the return value is *True,* else the return value is *False*. Call the Windows *GetLastError* function to get the error code.

Remarks

This function will change the address of the specified unit in the library records for all future attempts to send a message to the unit. The new value will be in place until a message, which originated from a different address but was from the same unit, was received by the library.

3.1.7 *GetUnitReport*

This function will return statistical information regarding the specified unit.

```
BOOL GetUnitReport
                (
         PUnitEntry  UnitEntry
                 );
```

Parameters

UnitEntry
Pointer to the PUnitEntry structure
Typedef struct PUnitEntry
{

```
UINT            StrucSize      // = 88
UINT            UnitNum
UINT            LastRxMsgNum
UINT            LastTxMsgNum
UINT            LastMsgRetrans
UINT            Address
UINT            RxMsgCiunt
UINT            TxMsgCount
UINT            SuceedTxMsgCount
UINT            AddressChangeCount
SYSTEMTIME      LastAddrChangeTime
SYSTEMTIME      LastAddrUpdateTime
SYSTEMTIME      LastMsgTime
          }
```

While the library is filling the data in the structure the *UnitNum* must be specified by the application prior to the call for this function otherwise the function will fail. SYSTEMTIME is a structure defined by Windows.

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call the Windows *GetLastError* function to get the error code. **Do not** try to use the data if the function fails.

Remarks

This function gives access to the library's internal database. Use it whenever you need information about a specific unit.

3.1.8 *GetIncomingBufferSize*

This function will return the statistic length of the incoming message waiting in the buffer at the specified index.

```
INT GetIncomingBufferSize
              (
              DWORD MessageIndex
                 );
```

Parameters

MessageIndex

Specifies the index of the required message in the incoming messages buffer. If the index is set to MESSAGES_COUNT then the return value is the amount of messages in the buffer.

Return Value

If the function succeeds the return value is the length of the desired message. The function will return -1 if it fails.

Remarks

This function can be called periodically to check for new messages. In this way you can avoid using the Windows messages mechanism.

3.1.9 *GetLibraryVersion*

This function will return the library version as 4 byte compressed data.

```
INT GetLibraryVersion (VOID);
```

Parameters

None

Return Value

The return value is a 4 bytes compressed version of the lowest byte representing the lower version byte (e.g. value of 67305985 which is 0x4030201, is in fact a version of 4.3.2.1). If the function fails the return value is null. Call *GetLastError* for the error code.

Remarks

This function can be called even if the library was not initialized.

3.1.10 *SetLibraryFlags*

This function will return the library version as 4 byte compressed data.

```
BOOL SetLibraryFlags (
DWORD    Flags
               );
```

Parameters

Flags - See the *InitializeLibrary* function for details.

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call *GetLastError* for the error reason.

Remarks

This function will change the library settings without the need to reset it. Not all flags are changeable without a reset, such as the UDP server control.

3.1.11 *AddToServerList*

This function will add an address to the allowed service addresses list.

```
BOOL AddToServerList (
LPCTSTR  Address
   );
```

Parameters

Address

A null terminated string that specifies an address to be added. The address must be a dotted IP string.

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call *GetLastError* for the error reason.

Remarks

The allow service list enables the library to serve as a software NAT for UDP messages. Any request by an application to the units sent to the host machine will be routed to the appropriate unit if the following conditions apply:

- The unit's address it already in the library database.
- The remote application is on the allow service list or the allow service to all flag is set.

3.1.12 *RemoveFromServerList*

This function will remove an address from the allowed service addresses list.

```
BOOL RemoveFromServerList (
LPCTSTR  Address
                                );
```

Parameters

Address

A null terminated string that specifies an address to be removed. The address must be a dotted IP string.

Return Value

If the function succeeds the return value is *True*, else the return value is *False*. Call *GetLastError* for the error reason.

Remarks

The allow service list enables the library to serve as a software NAT for the UDP messages. Any request by an application to the units sent to the host machine will be routed to the appropriate unit if the following conditions apply:

- The unit's address it already in the library database.
- The remote application is on the allow service list or the allow service to all flag is set.

3.1.13 *GetConnectedUnit*

This function will remove an address from the allowed service addresses list.

```
INT      GetConnectedUnit (
UINT*    Units
INT            ArrSize
                          );
```

Parameters

Units

Pointer to an UINT. This array will hold the units number currently known to the library. Set this parameter to null to specify that you only want the units count (See the Return Value section).

ArrSize

The size of the array.

Return Value

The return value is the amount of units known to the library.
If the function fails it will return -1. Call *GetLastError* for the error reason.

Remarks

Use this function in parallel to the *GetUnitReport* to get information about the units from the library.

4 Windows Messages

The library will send Windows messages to the application as part of its a-synchronic communication specification. The Windows messages do not have a constant ID. The library registers the messages every time it is loaded. According to Windows the values will not change until the next Windows session.

In order to get the current message values, call the Windows API function *RegisterWindowMessage* for each message.

The specified messages are listed in the following table:

Message Name	wParam	lParam	Details
GLM_UNIT_MSG_RECEIVED	Message length	Unit number	Sent every time a new message is received. When an application receives this message it should call the *PeekUnitMessage* function to retrieve the unit message.
GLM_MSG_TRANSMISSION_RESULT	Transmission result	Unit number	Sent for every call to the *PostUnitMessage* function to indicate the transmission result.
GLM_NEW_UNIT_FOUND	Null	Unit number	Sent every time a message is received from a unit for the first time.
GLM_UNIT_CHANGED_ADDRESS	Null	Unit Number	Sent when the library detects that the unit has a different address from the one in the database.

5 Constants

The following values are used by the library functions. For error codes, refer to the following *Error Codes* section.

VERIFY_HEADER	0x00000001
VERIFY_CHECKSUM	0x00000002
LOAD_UNIT_LIST_FROM_FILE	0x00000004
SAVE_UNIT_LIST_TO_FILE	0x00000008
NO_UDP_SERVER	0x00000010
REPLY_TO_INTEROGATOR_ONLY	0x00000020
ALLOW_SERVICE_TO_ALL_CLIENTS	0x00000040
MESSAGES_COUNT	0xFFFFFFFF

6 Error Codes

When a function fails it leaves an error code in the Windows kernel, specifying the failure reason. To retrieve the error code, call the *GetLastError* function from the Windows API.

Note that the *GetLastError* function is thread sensitive, meaning that the same thread that called the library function that failed must be the one that calls the *GetLastError*.

0x20000001	Unknown library error
0x20000002	Library was not initialized
0x20000003	Library already running
0x20000004	Unit does not reply
0x20000005	Unit is not yet known
0x20000006	No such message exists
0x20000007	UDP server already running on this machine
0x20000008	Address already Exist
0x20000009	Invalid index
0x2000000A	Invalid structure size
0x2000000B	Address not in the list

7 GPRS Manager Configuration Parameters

To define configuration parameters, open the GPRSManagerSetting.ini file, located under the IniFile folder. The following table describes how to edit the file and configure operational parameters.

Key	Valid value	Default	Description
TimeOut	Integer	8	Defines the time out 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, and 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 a unit did not transmit any data in this time 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 period in minutes.
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 it. Note that this log should be open ONLY for debug, since it will load the Linker once enabled.

Key	Valid value	Default	Description
Save Error Log	0/1	0	When set to '1' saves 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 unit windows, 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, and 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.