OLED Communicator

User Documentation

14 January 2021

What is the OLED Communicator?

The OLED Communicator enables a Personal Computer to Send and Receive SMS Text Messages without requiring any external internet connection. These SMS Messages can be used to provide notification of system/application events (outbound) as well as to receive custom command/control messages which can remotely direct processing or activities (inbound). "The Communicator" also has the ability to display informational messages on the OLED front panel. The Communicator is installed in a HotSwap Removable USB Drive Tray.

The OLED Communicator may be controlled via the Command Line, Batch Files, Custom Software (via an included AutoIT API Library), Serial Communication, or Network Communication. The OLED Communicator may even be used by other machines on the local network if so desired (via Network Protocols).

Unique Capabilities of the OLED Communicator

- The Communicator Sends/Receives SMS Messages directly via Cellular Network. No Internet connectivity is required. You can send (and optionally receive) SMS text messages from highly secure, air-gapped or fully firewalled, computer systems. Outbound SMS messages can be used to provide information about the status of programs or events. Inbound SMS messages can be processed for command/control of programs or activities.

- The Communicator provides a 16 char x 2 line OLED front panel display which can be used independently of its SMS Communication capabilities. Users and programs can easily display informational messages on the front panel of the PC for a specified duration or indefinitely (until replaced). The Communicator also automatically displays brief information on the OLED display any time an SMS message is sent or received.

- In the interest of security, there is no pre-programmed processing for inbound SMS text messages (just a visual display). Even this can be easily suppressed, and the device limited to SMS output only if desired. The user is 100% in control of exactly how input/output SMS messages should be handled. Inbound SMS messages can also be easily filtered and/or processed based on the caller's phone number

- The Communicator is completely User Programmable. For simplest access, the device can be controlled via the command line or batch files. To enable more sophisticated processing, an AutoIT API Library is also provided. (AutoIT is an easy to use, extremely capable, free development environment which can generate scripts or complied executables: https://www.autoitscript.com/site/). The OLEDCommAgent (driver), OCHelper, default SMSHandler, and the Example Programs were written using AutoIT and the OLEDCommSupport API. Several example programs have also been provided to demonstrate how simple it is to incorporate OLED Communicator functionality in user programs or batch files.

- The OLED Communicator is capable of displaying real-time information on the OLED Display when information is not actively being displayed by client applications. Custom “Idle Screen Handlers” can display real-time information such as CPU Usage, Disk Free Space, or a Stock Market Ticker.

- The Communicator can also be accessed/controlled via local USB Serial Protocols. The Communicator Agent (Driver) can be accessed/controlled via Network TCP Protocols (on a user-defined port) by both local programs and remote programs if so desired.

- The OLED Communicator is installed in a Hot-Swap Removable USB Drive Bay. It can be used and shared with any PC which uses “Digital Intelligence” Hot-Swap USB drive bays.

- Haute Solutions charges no monthly fees to use the OLED Communicator. You may use a GSM compatible SIM card from the provider of your choice with a service plan of your choice. There are multiple GSM cellular providers who offer pay-as-you-go texting service and do not charge a monthly fee.

Introduction to the OLED Communicator

The Communicator is housed in a USB Removable Drive Tray.

This is a photo of the front of an OLED Communicator as it might be installed in a PC. Note that the antenna is threaded and should be removed from the drive tray prior to insertion or removal from the system. The antenna will perform well with the threaded connection seated finger-tight.

The photo on the left shows the Communicator drive tray with the cover removed, it's important to note the location of the SIM card slot to the left of the SIM5320 Chip.

The photo on the right shows a partially installed SIM card to better illustrate the proper orientation for insertion. Note that the SIM card is installed with contacts down and the "notch" towards the display end of the tray, and to the outside of the PCB. The SIM card will "click" into the slot when fully and properly inserted. (Push it in again to "click" it and remove it)

Preparing for GSM/Cellular Functionality

In order to use the SMS capabilities of the Communicator, you will need to establish an account with a GSM Carrier and obtain a SIM card. Since the traffic requirements are very low for this device, we recommend a "Pay as you go" type cell service for the device. There are several Cellular providers which offer a very cost effective "Pay as you go" service without any monthly fee. The SIM5320A based OLED Communicator is a Quad Band 3G GSM device optimized for North American cellular frequencies. . It has the following communications capabilities:

- Supported Frequencies: 850/900/1800/1900
- Supported Protocols: EGSM/DCSI/PCSI/WDCMA/HSPDA

If you decide to enable GSM services on the device, you can obtain a SIM card from any 3G GSM provider. The Communicator requires the middle size "Micro" SIM card. Not the larger "Standard" SIM or the smaller "Nano" SIM card.

The registration process for each GSM carrier will be slightly different. However, you must register with your provider in order to obtain a SIM card and have a unique phone number associated with your Communicator.

Please note that some cellular providers (T-Mobile for instance) will require that you enter a validation code sent to your device in order to complete Pre-Paid Account Registration. If this is the case, you may want to have your Communicator fully installed (Hardware, Software, SIM Card) BEFORE you begin the registration process. The default/sample SMSHandler (SMSHandler.exe) will automatically pop-up a window on your PC which will display any inbound SMS Messages and provide an input field to send a response (if necessary

Alternatively, If you have an unlocked GSM cell phone, you may wish to temporarily install your SIM card into your phone to ensure that service has been properly established by your provider prior to installing the SIM into the Communicator. Send/Receive a few SMS text messages to confirm SIM registration and activation if you have the opportunity to do this.

Installing the OLED Communicator Software

Running the *OC_Install.exe* program will automatically install and configure the Communicator Software, Example Programs, and Source code for the device. The software will be installed (by default) into a *C:\Program Files\Haute Solutions\OLED Communicator* folder. A registry entry (Typically: "*HKEY_LOCAL_MACHINE\software\wow6432node\microsoft\windows\currentversion\run\OLEDCommAgent*") will be created to auto start the Agent/Driver (*OLEDCommAgent.exe*). The Agent/Driver will be automatically executed at the next reboot of the system (or can be started manually). However, a few final configuration settings will need to be made manually before starting the Agent for the first time…

Configuring the OLED Communicator Software

All of the configuration options for the Communicator are set in the *OLEDComm.ini* file located in the product directory (*C:\Program Files\Haute Solutions\OLED Communicator* by default). It is essential that you identify the associated COM port being used by the Communicator in order for the software to function properly.

Identifying the COM Port (Mandatory): The Communicator uses USB Serial Communication for communication between the hardware and the Agent (Driver). This will likely be different (but persistent) on each PC where the Communicator is used. The COM Port should be specified in by editing the "ComPort" setting in the "[USB]" section of the *OLEDComm.ini* file.

The simplest way to determine the COM port assigned to the Communicator is to open the windows "Device Manager" and observe the Com Port which is added when the device is inserted and turned on.

- Ensure that the Communicator is not physically installed and/or powered off
- Type "Device Manager" into the Windows Search Box, select the app to run it.
- Double Click on "Ports (COM & LPT)" to expand and note its contents
- Insert and turn on the Communicator and note the name/number of the COM Port which appears. It should be described as "USB Serial Port".
- Edit the OLEDComm.ini file to reflect the COM Port which was assigned to the Communicator

Configuring the TCP Port and Address (Optional): Applications generally communicate with the Communicator via a TCP Port. The default communications port is 65432. If you wish t5o change this port to something unique, simply edit the corresponding "IPAddress" setting in the "[TCP]" section of the *OLEDComm.ini* file.

LOCAL Applications will always use the TCP Loopback address (127.0.0.1). However, it is possible to communicate with a Communicator's Agent from remote machines on the network by specifying the IP address of the machine which has the Communicator installed. If you wish to use the Communicator from remote machines, be sure to change this IP address to identify the PC hosting the Communicator. You will likely also have to open up the firewall port on any machine hosting a Communicator if you wish to allow remote access.

Identifying an SMSHandler (Optional): By default, a very simple SMSHandler (*SMSHandler.exe*) is installed and configured to handle inbound SMS messages. The default handler will simply pop-up a dialog box to display any inbound SMS messages and accept a reply to send back (if desired). For security reasons, the default SMSHandler is NOT configured to act on or process any inbound messages (other than simply displaying them). If you wish to implement a more powerful SMSHandler, you will need to edit the corresponding "SMSHandler" setting in the "[Function]" section of the *OLEDComm.ini* file. Detailed information on how to design a Batch File or simple executable to handle inbound SMS Messages can be found in the "Basic Operation" section of this manual.

Identifying an IdleScreenHandler (Optional): By default, the internal "Idle Screen Handler" of the Communicator, will display the time on the front panel when no other "Active" message is being displayed. It is quite simple to implement a custom IdleScreenHandler to show some other information (CPU Usage, Disk Free Space, Stock Market Ticker) by specifying the application (or batch file) which will provide this information by specifying an "IDLEScreenHandler" setting in the "[Function]" section of the *OLEDComm.ini* file. The update period (in seconds) can also be specified by editing the "IdleScreenUpdate" setting in the same section. For more information in how to design an IdleScreenHandler, please see the "Advanced Operation" section of this manual.

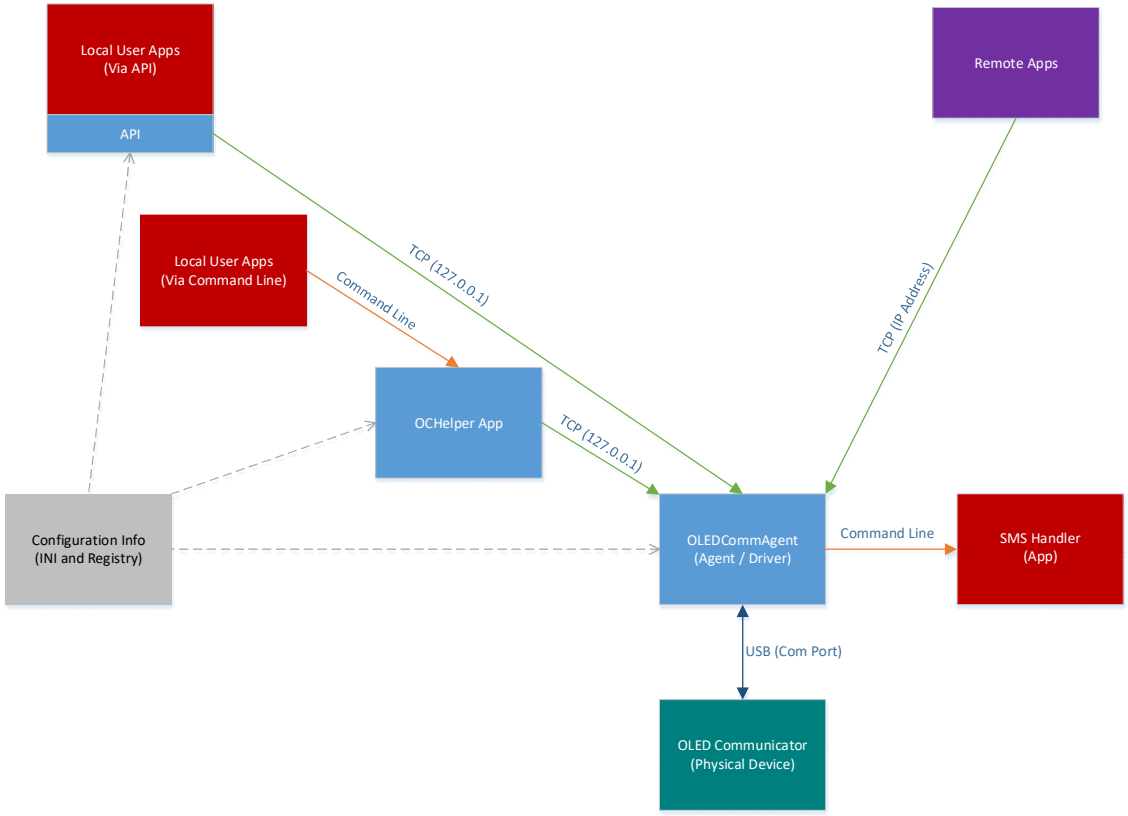
Allowing a "Slave Agent": By default, the OLEDCommAgent is expected to always be running. This way, any incoming unsolicited SMS messages can be forwarded to the designated SMSHandler. It is also possible to allow the OLEDCommAgent to be run on demand by client applications. Setting the "AllowSlaveAgent=1" in the "[Function]" section of the OLEDComm.ini file will allow "API" client applications to attempt to start the Agent if necessary. However, if an application starts the Agent, it will also terminate the Agent when the app terminates. It is suggested that you always start the Agent at system boot time and do not rely on the AllowSlaveAgent setting - particularly if you want to receive unsolicited inbound SMS messages!

Agent Startup Timeout (Optional): By default, the Agent will automatically display an error if it cannot establish connection to the Communicator within a specified time. This time may need to be adjusted based on the speed of your workstation, the number and complexity of the apps and services which are loaded at startup, and the order in which the Agent and other apps are started. You may adjust the startup timeout (secs) by editing the corresponding "AgentStartTimeout" setting in the "[Timeouts]" section of the *OLEDComm.ini* file. If you find that the agent is having trouble finding the Communicator at system startup, try extending this timeout (and be sure you have your COM port properly identified)

Suppressing Agent Startup Errors (Suggested): By default, the Agent will automatically display an error if it cannot establish connection to the Communicator within a specified time. Since the OLED Communicator is a removable device, it may not always be present in the system (when the Agent auto-starts). Once you are sure you have your "AgentStartTimeout" set properly, you may want to suppress Agent Startup errors if the device is not to be permanently installed in your system. Setting "StartupErrors=0" in the "[Config]" section of the *OLEDComm.ini* file will suppress Agent Startup errors and allow the Agent to terminate quietly. (The Agent can be started manually if a Communicator is inserted after a machine has already been started).

OLED Communicator Software Architecture

The following is a basic diagram of the software architecture and protocols used by the OLED Communicator:



OLEDCommAgent: Basic functionality of the OLED Communicator is handled via an Agent (driver). The Agent (*OLEDCommAgent.exe*) communicates with the OLED Communicator hardware via serial protocols over a USB connection. The Agent acts as a message and protocol handler for the Communicator. Generally, all communications to/from the Communicator should be handled by the Agent!

The Agent listens for inbound commands (from applications) on a standard TCP Port and then forwards them to the Communicator. Applications can issue commands to send SMS messages or display text on the OLED screen.

SMSHandler: The Agent polls the Communicator for inbound SMS Messages and then forwards them to the designated "SMSHandler" via the command line. As inbound SMS messages can come from any source, in any sequence, for any reason, a separate SMSHandler should be used to process the associated content.

TCP Communications: Local applications communicate with the Agent by using the TCP loopback address (127.0.0.1). Remote applications, running on network attached machines, communicate with the Agent using the host machine's public IP address. (If you wish to allow remote access to a Communicator, please ensure that any local firewall will allow traffic through the associated TCP port)

Local Applications (via Command Line): A helper application (OCHelper.exe) is provided to simplify command handling from local applications. The OCHelper app takes commands issues via the command-line and forward them via TCP to the Agent. The OCHelper provides a mechanism for non-programmers to issue commands to the Communicator via the interactive command-line or Batch/Cmd files. See the "Basic Operation" section of this manual for more information.

Local Applications (via API): If more sophisticated processing is required, an API (Application Programmer Interface) is provided for more advanced developers. This API is provided (in full source form) as an AutoIT Library. AutoIT is an easy to use, extremely capable, free development environment which can generate scripts or complied executables: https://www.autoitscript.com/site/). The OLEDCommAgent (driver) was written using AutoIT. The OCHelper and default SMSHandler were written using AutoIT and the OLEDCommSupport API. Source code for the OCHelper, default SMSHandler and several example programs is provided to illustrate how easy it is to use AutoIT and the API to generate applications which can use the Communicator.

Configuration Info: Although all of the Agent Configuration is maintained in a local INI file, there are several pieces of information which are maintained in the Registry to identify the location of the "master" INI file as well as the name and location of the currently running agent. The registry information enables API applications to locate the single/master configuration INI file. API User Applications, the OCHelper app, and the Agent all process the configuration information in this single/master OLEDComm.ini file

Basic Operation: Interacting with the Communicator using the Command Line (Batch Files)

Sending Commands to the Communicator: The OCHelper App will accept command-line parameters and pass them onto the Agent via the designated TCP port. Basic command line syntax for the OCHelper app is as follows:

OCHELPER [DISPLAY | SMS_SEND] [Params....]

Where: Function = DISPLAY [Line1] {Line2} {/TimeOut}

Line1: 1st Line of Msg to Display on OLED

Line2: 2nd Line to Msg (optional)

/Timeout: Secs to Display (optional) (/O = forever, default = /5)

Function = SMS_SEND [Number] [Line1] {Line2} {Line3}...

Number: Number to send SMS to

Line1: 1st Line of Msg

Line2,3,etc: 2nd/3rd/... Line of Msg (optional)

Here are some Examples:

- OCHELPER DISPLAY "This is Line1" "and Line2" /10

  Will display “This is Line 1” on the upper line and “and Line2” on the lower line of the OLED display for 10 seconds. (If you want this message to display forever (or until replaced), simply use “/0” as the final parameter).

- OCHELPER SMS_SEND +15551234567 "Hello World!" "How Are You?”

  Will send a 2 line SMS text message to +1-555-123-4567 consisting of “Hello World!” as the first line and “How Are You?” as the second line. Most cellular providers do not require a country code, so you can also likely just use the number with area code (if in the same country): “5551234567”. You can provide as few or as many lines as desired for your message.

Please note (for both DISPLAY and SMS_SEND Functions): You must enclose each “line” parameter in quotes in order to preserve internal spaces. (Standard rules of command line processing apply).

You will also likely have to specify the full path name of the OCHelper app in order for your commands or batch files to locate the application. (You could also put it somewhere in your existing path, or update your path to include its location). If the software is installed in the default location, the full path name of the OCHelper app is “*C:\Program Files\Haute Solutions\OLED Communicator\OCHelper.exe*”

Two Batch file examples are provided to demonstrate sending commands to the Communicator. *SendSMS.bat* and *DisplayMessage.bat* can be found in the *Examples**Batch* folder under the OLED Communicator installation folder (*C:**Program Files**Haute Solutions**OLED Communicator* by default).

Receiving SMS Messages from the Communicator: When the OLED Communicator receives SMS messages, it does not process them internally. All inbound SMS messages are sent to a user designated SMS Handler for processing. For security reasons the user is responsible for determining if and how inbound SMS messages should be processed. This process can be as simple or as sophisticated as desired.

The OLEDComm Agent will accept all inbound SMS messages and then forward them to the designated SMS Handler via the command line. (See the "Configuring the OLED Communicator Software" section in this document for info on how to designate an SMS Handler). The command line passed to the designated SMS Handler is formatted as follows:

1st Parameter:	{SMS Sender Number}
2nd Parameter:	{Line 1 of SMS Message}
3rd, 4th, 5th, .. Parameters:	{Line2, 3, 4,… of SMS Message}

The first parameter supplied to the handler is the cell number of the sender. The rest of the parameters are simply the message provided line by line. Batch, Cmd (powershell), or Executable files may be identified as the SMS Handler.

A sample Batch file examples is provided to demonstrate how a batch file might be used to receive and process incoming SMS messages. *SMSHandler.bat* can be found in the *Examples**Batch* folder under the OLED Communicator installation folder (*C:**Program Files**Haute Solutions**OLED Communicator* by default). The example simply prints out the incoming SMS Message to a text file.

More sophisticated Executables can also be developed to serve as an SMS Handler. These executables can be developed in any programming environment or any language as long as they can accept, parse, and process the information provided to them on the command line. (The OLEDCommSupport API is NOT required unless a reply to the SMS message is required – and even then, the OCHelper app could still be called to do this…)

It is entirely up to the user to decide what functions an incoming SMS message might perform and how to implement the associated security. The first parameter (sender SMS number) might be compared to only process communication from specified senders (although phone numbers can be spoofed). A second parameter (1st line in the SMS Message) might have to be a specific password. A third parameter (2nd line in the SMS message) might be a unique command/function to process. An SMSHandler could easily be designed to start processes, shut down systems, send back system or environmental information, or even wipe drives in a high security environment. An properly designed SMS handler can provide remote communication (command/control) of a system with no internet connectivity from virtually anywhere in the world. Implement your security wisely and choose your functionality carefully!

Advanced Operation: Programming Applications which use the Communicator API (AutoIT Library)

Several Examples are provided which demonstrate how applications may be developed using AutoIT and the supplied OLEDCommSupport library. These samples can be found in the *\Examples\API* folder under the OLED Communicator installation folder (*C:\Program Files\Haute Solutions\OLED Communicator* by default).

- AppWatcher: Launches an application and then sends out an SMS notification when the app terminates.

- IdleScreenHandler: Displays a Random Number on the OLED screen every time it is called (as determined by the IdleScreenUpdate setting in the OLEDComm.ini file)

- SMSClient: Sends an SMS text message to the specified number

- TextClient: Displays a message on the OLED Communicator display

AutoIT source code for the OLED Communicator software components is also provided (OCHelper.exe, SMSHandler.exe, and the even the OLEDCommAgent.exe itself).

Designing programs in the AutoIT environment which use the OLED Communicator is fairly simple:

- Always include the "OLEDCommSupport.au3" library at the beginning of your script/executable.

- Always shut down the OLEDSupport services before exiting your script/executable using the "OLEDComm_Shutdown()" function. This is particularly important if you are using a "Slave Agent" (see "AllowSlaveAgent" in the Software Configuration section of this document.

- Use the "OLEDComm_Display()" function to display information on the OLED screen. The OLEDComm_Display function takes three arguments: First Line (String), Second Line (String), and Timeout (Number). The Timeout parameter specifies the number of seconds to keep messages on the display. A Timeout value of "0" (zero) leaves the message on the screen indefinitely (until replaced).

- Use the "OLEDComm_SendSMS()" function to send SMS messages. The OLEDComm_SendSMS function takes two arguments: SMS Number to send message to (string), and the Message itself (string). The Message string can contain embedded spaces, line feeds, carriage returns, etc.

Please examine the provided example scripts/programs for more information.

Advanced Operation: Implementing a custom IdleScreenHandler

If no IdleScreenHandler is specified, the OLED Communicator will display the current system time on its screen. If you wish to display something "more interesting" on an idle display (CPU Usage, Disk Free Space, Stock Market Ticker), then a provision is available to implement a custom "IdleScreenHandler". (See the "Identifying an IdleScreenHandler" in the "Configuring the OLED Communicator Software" section of this document).

An IdleScreenHandler is simply an application which is called at a predetermined interval (when no other application is using the Display). The interval is set in the *OLEDComm.ini* file by using the "IdleScreenUpdate" parameter (seconds) in the "[Function]" section.

The Agent will call (run) the IdleScreenHandler application at the specified interval. Only one parameter is provided on the command-line when executing the IdleScreenHandler. The value of the IdleScreenUpdate setting is provided as the sole parameter as some handlers may want to know how often they are being executed.

It is expected that the IdleScreenHandler will then use the API OLEDComm_Display() function (see previous section) to place some information on the OLED Display. It's that simple!

Troubleshooting

The OLED Display on the Communicator can be a useful source of diagnostic information. Diagnostic messages are generated both at boot time and as SMS Messages are sent and received.

Normal Boot Messages: The OLED display on the communicator will report its progress as it boots up and comes online. The following is the basic sequence of information as it is displayed:

At initial startup, the Communicator will report its firmware version:

OLEDCommunicator
OLEDCOmm FW V1.0

The communicator will then attempt to establish a connection to the cellular network:

OLEDCommunicator
Starting GSM…

The Communicator will report the following once a connection to the Cellular Network is established (or has timed out). This may take up to 60 seconds or longer:

OLEDCommunicator
GSM Initialized

If the Communicator detects that NO SIM CARD is installed (or inserted improperly), the following message will BRIEFLY be displayed. (It is possible to operate the OLED Communicator without cellular connectivity to only display messages) If you think you might be having connectivity issues, please restart the OLED Communicator and watch for this message to determine if your SIM card is properly inserted:

OLEDCommunicator
No SIM Inserted

The Communicator will display the following message when it has fully booted and is waiting for the Agent to connect:

OLEDCommunicator
No Agent

Once the Agent connects, the following message will be briefly displayed:

OLEDCommunicator
Agent Connected

After the "Agent Connected" message times out, the Agent will run it's default Idle Screen Handler to display the system time:

OLEDCommunicator
{Time}

SMS Status Messages: Whenever an SMS message is sent or received, the event will be reported on the OLED display for 5 seconds.

Inbound SMS Messages: When receiving an SMS message, the first line of the OLED Display will report "IN:" followed by the cell number of the sender (up to 16 characters total). The second line will display the first 16 characters of the inbound message

Outbound SMS Messages: When sending an SMS message, the first line of the OLED Display will report "OUT:" followed by the cell number of the intended recipient (up to 16 characters total). The second line will display the first 16 characters of the outbound message.