intelligence everywhere™

Motorola TxMessenger®
Installation and System Administration Guide

For release with version 3.5. November 21, 2003

Table of Contents

1 Introduction

1.1 Warranty Disclaimer

Motorola may add, delete, change, or withdraw, in whole or in part, this document or the equipment or software or the specification described in this document at any time and without notice. Such document modifications will be incorporated in new releases of this document on an intermittent basis.

This publication could contain technical inaccuracies or typographical errors. Motorola disclaims any responsibility for labor or material cost involved by persons outside the company as a result of using this document.

MOTOROLA, INC., PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND EITHER EXPRESS OR IMPLIED INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MOTOROLA SHALL NOT BE LIABLE FOR ANY DAMAGES (INCLUDING, BUT NOT LIMITED TO, CONSEQUENTIAL, INCIDENTAL, SPECIAL DAMAGES OR LOSS OF PROFITS) ARISING OUT OF OR RELATED IN ANY WAY TO THIS DOCUMENT OR ITS USE.

Features or capabilities described as "future" may not ever be implemented or available, and/or there may be additional costs or requirements. Experimental settings are not thoroughly tested and may not function correctly.

The flexibility and power offered by the scripting language and huge variety of settings opens opportunities for negative interactions or failures due to configuration. Motorola certifies primary features only, and reserves the right to decline corrections for secondary features or to require configuration changes that prioritize primary features even at the loss of a secondary feature.

1.2 Purpose

This document contains instructions and reference information for the Motorola TxMessenger wireless data mobile messaging software. An experienced technician or system administrator can use this guide to help prepare, install, and maintain TxMessenger.

This guide is not intended for end users or inexperienced personnel.

Information contained in the Motorola TxMessenger release notes supersedes this document.

1.3 Assumptions

The reader needs to be experienced with the operating system on which the software is going to be installed and maintained. If reading configuration information or copying files from a prior messaging application on Windows 3.1x or DOS, the reader is required to have a working knowledge of that operating system.

The operating systems must be properly installed on the various models of computers.

The reader needs to have a working knowledge of the computers to be installed on (Motorola MW-520, Panasonic CF-25/CF-27, or any other certified device) or upgraded from (such as a laptop or Motorola 9100). The computers must be in proper working order.

The reader should have a working knowledge of Motorola Private DataTAC™ wireless data systems and the particular customer's message switch. The reader may need experience with Motorola RSS (Radio Service Software) and Motorola PRM/VRM radio modems. The system, message switch, and radio modems must be in proper working order and programmed to operate together.

Items pertaining to RSS (Radio Service Software) and reprogramming radio modems are applicable only to qualified technicians, who are proficient in such tasks without requiring any instructions from this document.

The reader should have a basic understanding of messaging applications. If upgrading from Motorola TX and/or Motorola WaveSoft-Link messaging applications, those programs must be in working order and the reader needs a working knowledge of those programs.

The computers should be backed up and the proper software and other resources should be available in case the computer needs to be restored to its original configuration.

1.4 What Does "TX" Mean?

Motorola's original data mobile messaging application was named TX (Motorola TX or Motorola TX for Windows). The application was available on Motorola 7100, 9100, KDT480, and other Motorola devices, as well as available for laptops, Fortés, and MW-520s running the Microsoft Windows 3.1 operating system.

The TX application's file formats, structures, and over-the-air formats of forms and messages are commonly referred to as TX-format or the TX-protocol. This document uses the term "TX application" or "Motorola TX" to refer to the messaging application software. This document uses the term "TX-protocol" to refer to the over-the-air format made popular by the TX application.

The messaging application Motorola WaveSoft-Link used most of the TX-protocol (over-the-air). The headers of WaveSoft-Link's files are slightly different and therefore WaveSoft-Link can't read or write files that are compatible with the TX application.

The new messaging application, Motorola TxMessenger, also uses most of the TX-protocol (over-the-air). In addition, TxMessenger introduces some new over-the-air features. Some of the TX and WaveSoft-Link application file formats can be read directly by TxMessenger, although TxMessenger's file formats can't be read by TX or WaveSoft-Link.

Be careful not to use the term "TX" to refer to TxMessenger. TxMessenger is a completely new product, but can take advantage of most of the old TX-protocol and old Motorola TX application file formats.

1.5 Text, Graphics, and Screen Shots

A system administrator can modify most of the text and many of the pictures in the TxMessenger application. Therefore, the screen shots and button names provided in this document may not match a customized version.

Because of differences in operating systems and third-party drivers, Windows instructions and screen shots in this document may differ from those for the reader.

1.6 References

Most numbered documents are available from Motorola Americas Parts Division, 1-800-422-4210. The *Modem Interface Specification* has an excellent references section that lists additional documents and sources.

Document Number	Title
(TxMessenger)	*TxMessenger Settings* (spreadsheet)
(TxMessenger)	*TxMessenger Settings Conversion Chart* (spreadsheet)
68P04016C25	*TX for Windows Host Programming Guide*
68P81063C20	*TX Host Programming Guide*
68P02948C50-O	*TX Configuration Utility User's Guide*
68-81063C60-C	*FORMGEN TX-Forms Generator User's Guide*
68P81130E05	*WaveSoft-Link Installation and System Administration Guide*
DCS/APG 480P-A	*KDT-480 Product Description Notebook Terminal And Host Application Programming Guide.* (Probably out of print)
68P04014C30-0	*9100-386 Mobile Work Station Application Developer's Guide*
68P02950C70-O	*Mobile Workstation 520 Application Developer's Guide*
444.5025.001.200	*Modem Interface Specification*
68P02946C95	*VRM 500/600 Vehicular Radio Modem Radio Service Software User's Guide*
68P81063C20-B	*MDT/TX Terminal and Host Application Programming Guide.* (Probably out of print)

2 Product Requirements

 **Important:** As with any software program, failure to observe these minimum product requirements (or running out of memory or disk space during installation or operation) may result in reduced performance, partial or complete loss of functionality, and/or permanent data loss, including permanent loss of message(s).

2.1 Certified Computers

Motorola MW-520 (grayscale, 350 NIT color, 350 NIT color touch screen, 1000 NIT color)
Panasonic CF-25
Panasonic CF-27 (with and without touch screen)

TxMessenger is certified to operate on the above computers when those computers have been configured to comply with all the other product requirements.

TxMessenger doesn't support the Motorola Forté.

2.2 Operating System

Requires Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows NT Workstation 4.0, Microsoft Windows Me, Microsoft Windows 2000, or Microsoft Windows XP.

In all cases, the operating system should be virus free and maintained with the most recent, stable Microsoft service pack upgrades.

TxMessenger doesn't work on earlier versions of Microsoft's operating systems, including Windows 3.1, Window 3.11, Windows NT 3.51, and DOS. TxMessenger also doesn't work on other operating systems, including Windows CE, Linux, PalmOS, and MacOS.

2.3 Processor

Requires Intel Pentium 120 MHz or higher.

2.4 Memory

Requires 16 MB (megabytes) of RAM minimum.

This amount is suitable for most customers. However, any of the following may require additional memory:
- large number / size of messages stored
- large number / size of forms, pictures and/or sounds used

For operating system memory requirements, please check the Microsoft website.

2.5 Disk Space

Requires 16 MB (megabytes) minimum available.

This amount is suitable for most customers. However, any of the following may require additional space.
- large number / size of messages stored
- large number / size of forms, pictures and/or sounds used

2.6 Display

2.6.1 Display Size

Requires 640 x 480 or larger.

Form widths greater than 40 characters across may not be readable except on a full-size window or larger displays.

Pictures in forms or messages are not resized and therefore may be placed or lined-up differently on various-sized windows and displays.

2.6.2 Display Colors / Color Palette

Requires thousands of colors/gray scales or greater.

Some display driver's control panels may refer to Thousands as: 16-bit, 32768, 65536, or High. Better modes may be referred to as: 32-bit, Millions, or True.

MW-520s that have grayscale screens (also called B&W, black and white, or monochrome) can be used as long as the display's color palette is set to thousands or greater. Be sure to follow the steps in section 3.7 "Set the Display". Although TxMessenger detects the grayscale MW-520 screen and can automatically adjust certain settings, the administrator can also manually modify those settings in TxMessenger.ini for optimal contrast. For example, the font color looks better as black.

If TxMessenger is operated on a display set to less than thousands of colors (such as 2, 16, or 256 colors), the window images may be washed out and generally be unacceptable for use.

2.6.3 Font Size

Requires Small Fonts.

Text may be truncated, not visible, or illegible if the Large Fonts (or anything other than Small Fonts) display setting is selected.

2.6.4 Fonts

Requires Arial, Arial Bold, and Lucida Console.

Text may be truncated, not visible, or illegible if these fonts are not properly installed on the Windows operating system.

2.7 Input Device

Requires a keyboard and mouse (or mouse compatible pointing device such as a touchpad).

TxMessenger's large interface elements make it easy to use on mouse-compatible touch screens.

Mouse scroll wheels may not operate the TxMessenger scroll bars.

Handwriting and pen input-devices (such as on the Motorola Forté) aren't supported.

2.8 Modem

TxMessenger requires an internal Motorola Private DataTAC RF modem or one available RS-232 serial port for connection to an external Motorola Private DataTAC RF modem. Modem must fully meet all of the specifications listed in *Motorola Modem Interface Specification* 444.5025.001.200. TxMessenger is certified with external VRM500, VRM650, VRM660, and VRM850, as well as the internal MW-520 radio modem.

The PRM240 modem contains a code plug which is different than prior DataTAC modems. Due to differences in supported commands, this code base is not yet certified with TxMessenger. However, to attempt experimental use, see the settings that begin with "Modem" (in particular "ModemByWillo") in the *Motorola TxMessenger Settings* spreadsheet.

 **WARNING:** Cycling power (such as ignition sense) on the radio modem, disconnecting the cable, or undocking while any TX-protocol application (TX for Windows, WaveSoft-Link, TxMessenger, etc) is running may cause a message to be acknowledged to the host but not delivered to the user (message delay or message loss). For example, the bytes of the message could be in transit in the modem cable when you physically disconnect the cable. To improve proper modem initialization and message delivery, a TX-protocol application must always be exited before disconnecting (such as undocking) the modem. This is not an application or modem defect, it is an inherent limitation of the TX Protocol. TxMessenger and other applications must follow the TX-protocol to be compatible with existing TX-protocol hosts.

2.8.1 Serial Communications Port

COM1 to COM99

The communications port chosen must be an "IBM AT compatible" serial COM port, using type 16450 or 16550 UART and industry-standard port/interrupt assignments (port 3F8/IRQ4 or port 2F8/IRQ3).

While running, TxMessenger requires exclusive use of the COM port, just like any other application. To run any other communications software that needs access to the COM port, TxMessenger must be exited.

2.8.2 Baud Rate

110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 56000, 57600, 115200, 128000, or 256000

Both native and transparent presentations must be set to the same baud rate.

2.8.3 Flow Control

RTS/CTS (preferred), XON/XOFF, or none

2.8.4 Mode

Mode 1 or 4 is required.

Modes 2 and 3 aren't supported.

2.8.5 Modem Code Version

R01.19.02 or better

2.8.6 Native SDU and Offline AT Commands

TxMessenger makes use of both Native SDU commands and Offline AT commands. Therefore, a fully-compliant DataTAC modem is required.

2.9 Media

2.9.1 Factory Shipping Media

The complete TxMessenger software product (installer, manual, examples, picture collection, utilities, and so on) ships on a special type of CD-ROM, a read-only CD-R. A computer with CD-R read capability is required for the system administrator. Note, first generation DVD drives and some early CD-ROM drives are unable to read the type of CD-ROM on which TxMessenger is shipped.

The complete TxMessenger software product is also available on a PCMCIA ATA card.

2.9.2 Installation Media

The TxMessenger installer may be run from the factory media or may be copied and run from a network drive or removable media. The final disk of the installer may be modified to include: site-specific forms, messages, pictures, sounds, and configuration information. As such, it is suggested that the factory CD-ROM installer be copied onto writable media, such as floppy disks or a PCMCIA card.

The advantages of PCMCIA cards are that they are fast and can have a large enough capacity to hold a significant amount of site-specific files (forms, pictures, etc.). The disadvantages of PCMCIA cards are that they tend to be more costly, more delicate, and less universally compatible as are floppy disks. The disadvantages of floppy disks are that they are slower and can only contain as many site-specific files as there is capacity remaining on the final floppy disk.

Based on the installation media chosen, a floppy drive or other appropriate device is necessary on the destination computer.

3 Preparing For Installation

3.1 Suggested Installation Methodology

The generally suggested installation method is to fine-tune a single device before proceeding with the fleet. After a single device-type tests as fully operational, master disks are then created to copy the TxMessenger settings and customer-specific files during TxMessenger installation onto similar device-types.

In all cases:

- Make a backup of the destination computer and radio modem settings.
- Reboot after each installation or removal of software.

3.1.1 License Settings

With TxMessenger version 3.0 and higher, there are four license settings within the TxMessenger.ini file that must remain in the TxMessenger.ini during the installation of all the customer's devices. These four settings are:

```
LicenseCustomerName
LicenseExpirationDate
LicenseMaximumNumberOfClients
LicenseNumber
```

Simply enter the four license settings into the master TxMessenger.ini file, perform any other desired configuration, test it live, and then copy the ini file to the installer disks to deploy to a customer's fleet. These settings are for a site license, not a license that needs to be unique for each device.

If a user tries to launch TxMessenger without all four of these settings or if any of the settings get altered, you will see this error message.

If you receive a new version of TxMessenger and your maintenance plan has expired, you will need to contact your Motorola representative to renew your maintenance plan in order to receive updated license settings for the TxMessenger.ini file. One way to verify if you will need to renew your maintenance plan, is to check the date on TxMessenger.exe. Right-click TxMessenger.exe and click the version tab. Under item name on the left hand side, select Product Version. The right hand side will show the product version number and the product version date. If the product version date is newer than the date in the `LicenseExpirationDate` setting in your TxMessenger.ini file, then TxMessenger will not run until you obtain updated license settings. When you attempt to run TxMessenger under these conditions, you will see this message.

TxMessenger will continue to run after clicking OK on either of these alert windows, but the modem will be disabled and the user will not be able to re-enable the modem until the settings have been updated.

3.2 Copying to Installation Media

3.2.1 For Floppy Disks

The original factory media (CD-ROM or ATA card) contains a folder named “Installer (Disk Images)”. Within that folder are three folders named Disk1, Disk2, and Disk3.

Copy the contents of each folder (Disk1, Disk2, and Disk3) onto preformatted floppy disks. The contents of each folder should be copied to its own floppy disk.

Write the disk number and TxMessenger version on each disk label.

This should result in three disks, each with the contents of each folder from the factory media.

3.2.2 For Other Media

The original factory media (CD-ROM or ATA card) contains a folder named “Installer”.

Copy the Installer folder itself (including its contents) to the desired installation media.

This should result in one disk or card, with a folder named Installer.

3.3 Select a Sample Device

Choose a computer that is:

- Easy to access.
- Has convenient and appropriate media capability (floppy drive, etc.).
- Is known to be working and without viruses.
- Is the same as a group of destination devices in the rest of the fleet.
- If upgrading, contains a fully working version of the prior application software and fully working modem.

3.4 Exit All Applications

Quit any applications currently running on the destination computer, especially messaging applications like Motorola TX, WaveSoft-Link, or TxMessenger. Also, make sure to exit the messaging application's communication engines, such as WaveGuide (Motorola TX) or MCS (WaveSoft-Link).

TxMessenger requires dedicated access to the modem and other messaging applications may prevent TxMessenger from connecting (and vice-versa).

3.5 Make A Backup

If devices are going to be upgraded where a messaging application has already been installed, the customer specific settings and forms should be backed-up before beginning the software installation.

Making a complete backup of a computer before installing new software allows the computer to be restored to the previous state if the new installation fails to perform adequately. A complete backup is also useful for other reasons, such as hard disk failure.

However, a complete backup is usually impractical, in which case, at least make a backup copy of the following files and folders:

3.5.1 Motorola TxMessenger

```
\Program Files\Motorola TxMessenger\TxMessenger.ini
\Program Files\Motorola TxMessenger\Forms\      (entire folder)
\Program Files\Motorola TxMessenger\Messages\   (entire folder)
\Program Files\Motorola TxMessenger\Pictures\   (entire folder)
\Program Files\Motorola TxMessenger\Scripts\    (entire folder)
\Program Files\Motorola TxMessenger\Sounds\     (entire folder)
```

Note: The 'Scripts' folder was not available in versions prior to v2.5.

3.5.2 Motorola TX for Windows

```
\Txwin\Labels.pfk
\Txwin\Txwin.cfg
\Txwin\Forms\     (entire folder)
\Txwin\Labels\    (entire folder)
\Windows\Moto_sys.ini
```

Refer to the product's documentation to determine if there are any additional files that should be backed up.

3.5.3 Motorola WaveSoft-Link

```
\Motorola\Ws-link\Wslink.ini
\Motorola\Ws-link\Messages\   (entire folder)
\Motorola\Ws-link\Rcvfmts\    (entire folder)
\Motorola\Ws-link\Rsdntfmt\   (entire folder)
\Motorola\Common\Np_init.sys
\Windows\MCS.ini
```

Refer to the product's documentation to determine if there are any additional files that should be backed up.

3.6 Modem Settings

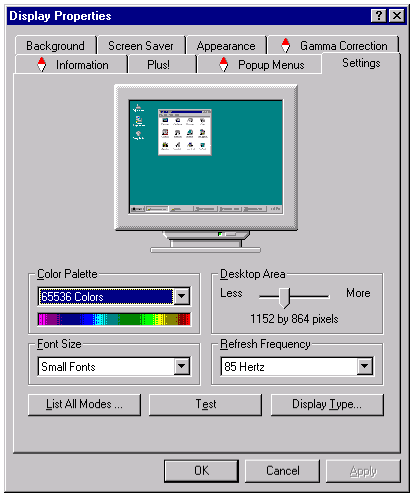
Using RSS (Radio Service Software), save or write down the current modem settings and modem software version for each device. Take steps to be able to restore the modem to the original settings if necessary.

Modify any of the existing modem settings if they don't match the modem requirements listed in section 2.8 of this document.

3.7 Set the Display

Most displays can have their settings configured as follows:

1. Choose Start→Settings→Control Panel from the Start menu on the Windows task bar.
2. In the Control Panel, choose Display.
3. In Display, choose the Settings tab.
4. In Settings, set the Color Palette to 32768, 65536, 16-bit, True color, or greater.
5. In Settings, set the Desktop Area or Screen Area to 640x480 or more.
6. In Settings (on Windows NT), set the Font Size to Small Fonts. In Windows 95/98, click the Advanced button to set the Font Size.
7. Click the OK button.
8. Restart the computer.



4 Installing

4.1 Disk and Operating System Security Rights

The installer creates directories, copies files, adds fonts and a startup file to the system, and modifies the registry. In Microsoft Windows NT, or any operating system with security or virus protection, it may be necessary to disable security/virus protection programs or logon with administrative rights to install TxMessenger successfully.

4.2 Running the Installer

On the destination computer, run Setup.exe from the installation media to begin the installation procedure.

4.2.1 For Floppy Disks

Insert the first disk into the destination computer and run Setup.exe. The installer will ask for the remaining disks as needed.

The location of the installer program on most floppy drives would be A:\Setup.exe. Some PCMCIA floppy drives, like on the MW-520, are located on D: instead of A:.

4.2.2 For Other Media

Insert the installation media into the destination computer and run Setup.exe from the \Motorola TxMessenger Installer\Disk1 folder. Because the remaining disks are in the same folder as Disk1, the installer finds them automatically as needed without asking.

If the installer displays an error message, crashes, or continually asks to swap between disks, it is likely that the installer was either not copied over correctly or that the media is bad. Try reformatting the media and recopying the installer.

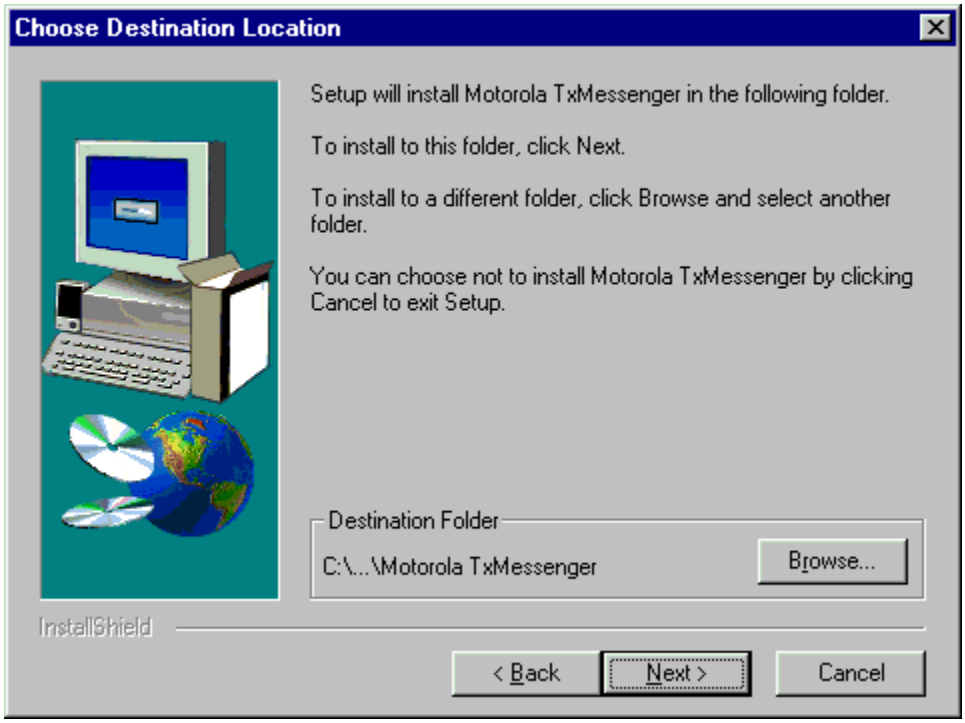
If the TxMessenger logo in the installer background appears washed out or unattractive, the computer's display is probably less than the required thousands of colors. Follow the instructions on changing the display colors as described in section 2.6 of this document.

4.3 Release Notes

The release notes contain more recent information than this document. Before continuing, read the release notes. The release notes are also available on Disk 1 as `Release Notes.txt`.

4.4 Destination Directory

`C:\Program Files\Motorola TxMessenger\` is the default installation path provided by the TxMessenger installer and is referred to throughout this document. Note that a different destination location can be chosen during installation of the TxMessenger software (or other software packages), in which case the reader should substitute their specific destination paths accordingly in all examples.

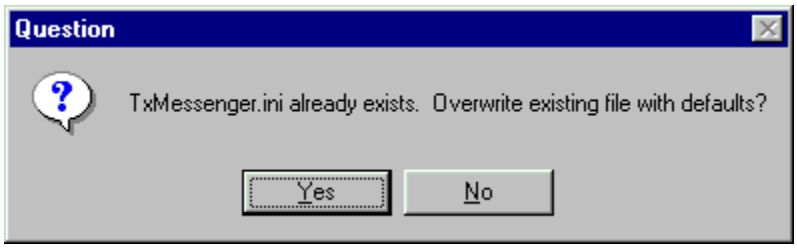


The destination directory shouldn't be changed unless necessary. It is easier to find, upgrade, and maintain TxMessenger if it is located in the default directory.

The same or various versions of TxMessenger can be installed to different directories or drives. Each copy of TxMessenger uses the settings, forms, messages, pictures, and sounds located in its individual directory. However, only one copy of TxMessenger can be run at the same time. The currently running copy of TxMessenger continues to run when the user attempts to launch another copy.

4.5 Overwriting an Existing TxMessenger Directory

If TxMessenger.ini already exists in the destination directory, the installer asks whether the existing TxMessenger.ini should be replaced.



If the TxMessenger.ini file is out of date or has become corrupted and unreadable, then it can be replaced by clicking Yes. Otherwise, the current settings can be retained by clicking No.

Beginning in version 1.2, the existing TxMessenger.ini file is backed up and renamed with a numeric filename extension (like TxMessenger.001) in the destination directory when the installer is told to overwrite the file.

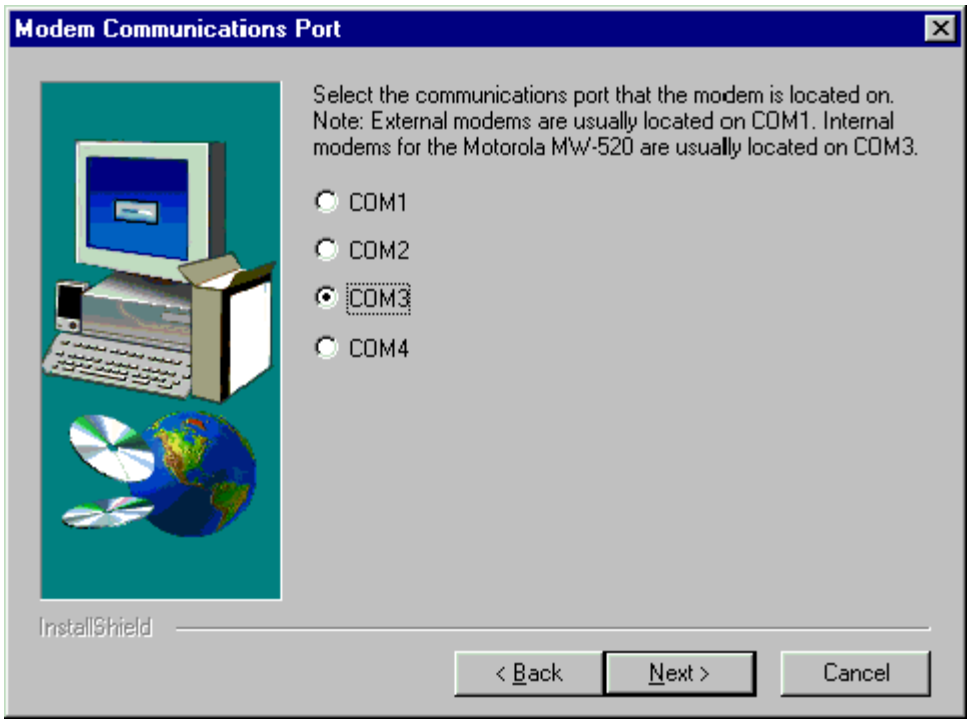
The contents of the Forms, Pictures, Messages, and Sounds folders on Disk 4 of the installer are always copied to the destination directory. The files on Disk 4 will replace the files of the identical names in the destination directory without asking, but files with different names in the destination directory will remain unchanged. Therefore before installation:

- manually delete files in the destination directory that are no longer desired.

- make a backup copy of files in the destination directory that should remain unmodified. After installation, manually copy those desired files back. This is only necessary for files in the destination directory that have the same name as files copied over by the installer. Files with different names are not overwritten or deleted by the installer.

4.6 Modem Communications Port

Choose the computer's communication port to which the radio modem is connected. Internal modems (such as in the MW-520) are usually located on COM3, while external modems are usually on COM1.

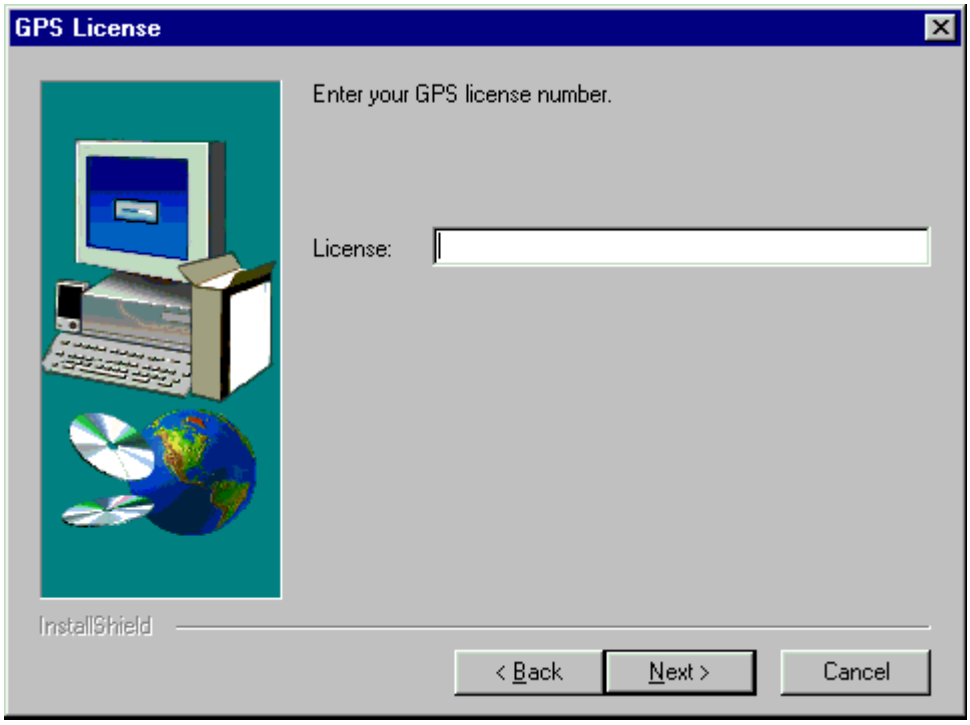


The installer reads the default or previously installed (or just overwritten) TxMessenger.ini to determine the default modem port to display. Any changes to this setting are always written to TxMessenger.ini in the destination directory, even if the user answered "no" to overwriting the existing settings with defaults.

To change the communication port after installation, set `ModemPort=#` in TxMessenger.ini, where # is a number from 1 to 99.

4.7 GPS License

To enable GPS, enter the GPS license number provided.

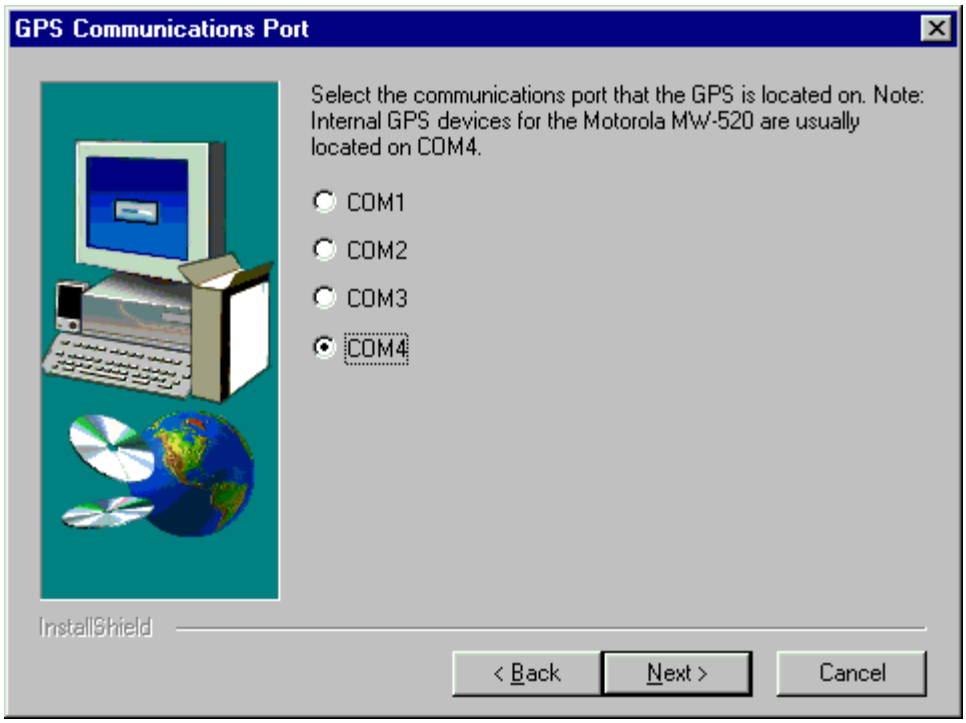


The installer reads the previously installed (or just overwritten) TxMessenger.ini to determine the default GPS license to display. Any changes to this setting are always written to TxMessenger.ini in the destination directory, even if the user answered “no” to overwriting the existing settings with defaults.

To change the GPS license after installation, set `GPSLicense=#` in TxMessenger.ini, where # is a valid GPS license number.

4.8 GPS Communications Port

Choose the computer's communication port to which the GPS device is connected. Internal GPS devices (such as in the MW-520) are usually located on COM4.

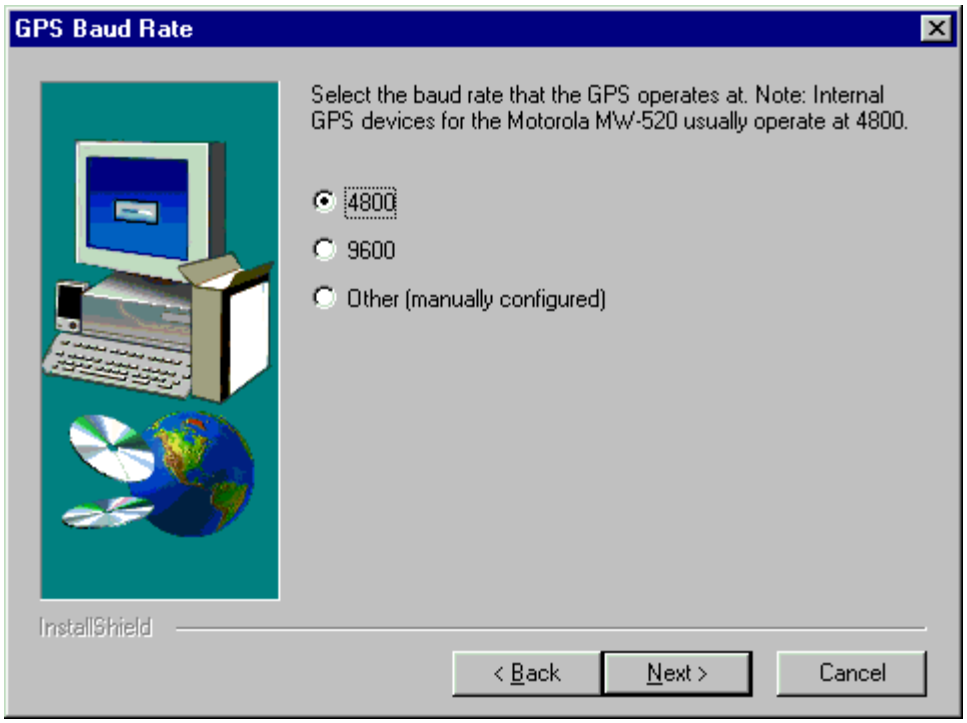


The installer reads the default or previously installed (or just overwritten) TxMessenger.ini to determine the default GPS port to display. Any changes to this setting are always written to TxMessenger.ini in the destination directory, even if the user answered "no" to overwriting the existing settings with defaults.

To change the communication port after installation, set `GPSPort=#` in TxMessenger.ini, where # is a number from 1 to 99.

4.9 GPS Baud Rate

Choose the baud rate at which the GPS device communicates. Internal GPS devices (such as in the MW-520) usually operate at 4800.

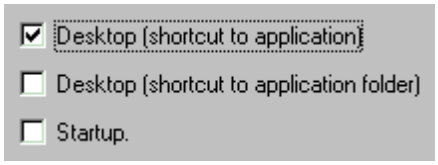


The installer reads the default or previously installed (or just overwritten) TxMessenger.ini to determine the default GPS baud rate to display. Any changes to this setting are always written to TxMessenger.ini in the destination directory, even if the user answered "no" to overwriting the existing settings with defaults.

To change the GPS baud rate after installation, set `GPSBaud=#` in TxMessenger.ini, where # is the baud rate.

4.10 Adding TxMessenger to the Desktop and Startup Group

The TxMessenger icon is always added to the Windows Start→Programs→Motorola TxMessenger Menu. In addition, near the end of the installation process, the installer provides the following additional choices:



- Adding the TxMessenger program icon to the Windows Desktop makes running TxMessenger more convenient for the user.

- Adding a TxMessenger application folder shortcut to the desktop makes it more convenient for maintenance.

- Adding TxMessenger to the Windows Startup Group makes TxMessenger run automatically every time Windows starts up.

  To effectively run TxMessenger from the Startup Group, any internal radio modem must be powered on early enough in the boot process so that it is ready when TxMessenger starts. On a Motorola MW-520 with internal modem, the AUTOEXEC.BAT must run MW_RON.EXE.

4.11 Restart the Computer after Each Install or Uninstall

Always restart the computer after any uninstall or install of software. If the computer is not restarted between uninstalling and installing, the installation may not be successful.

4.11.1 Font(s) Missing

If the computer hasn't been restarted, the required fonts may not be available. TxMessenger displays a warning whenever a required font isn't available:

If restarting doesn't correct the problem, check the Windows Font folder (Start→Settings→Control Panel then open Fonts) to make sure that the following fonts are installed:

Font Name	Filename
Arial	`Arial.ttf`
Arial Bold	`Arialbd.ttf`
Lucida Console	`Lucon.ttf`

If the fonts are installed, open the font in the Windows Font folder by double clicking the font file to view a sample of it. This usually causes the operating system to recognize the font and make it available to applications.

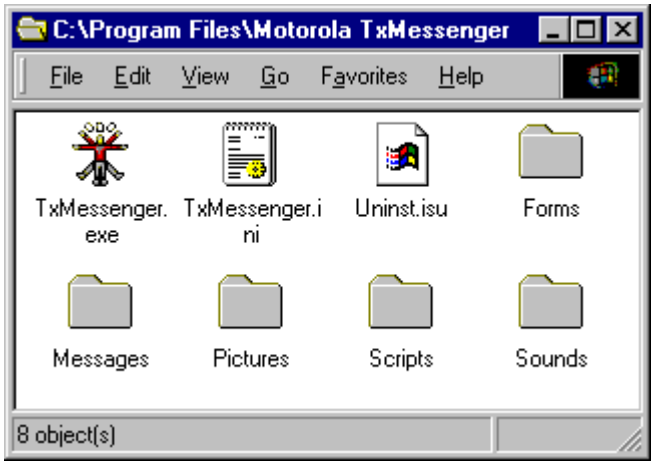
Setting `FontRequiredMissingText` to blank in TxMessenger.ini can disable the warning.

4.12 Creating A Site-Specific Master Installer

Creating a master installation disk set is the easiest way to install on multiple workstations, such as a mobile fleet. With a master installation disk set, TxMessenger can install itself along with customer-configured forms, pictures, sounds, and settings – all at the same time.

1. Follow the steps to prepare and install TxMessenger.
2. Select a single computer that represents a common operating system or hardware model.
3. Install TxMessenger on that computer.
4. Create or copy over any forms, sounds, and pictures.
5. Configure TxMessenger.ini.
6. Run TxMessenger and confirm that all forms, function keys, onscreen keys, status messages, coded messages, text, colors, and other features operate as desired over an active radio modem on a live system.
7. Exit TxMessenger.
8.  **Important:** Set Configure=FALSE in TxMessenger.ini. If Configure remains TRUE, the users will be able to:
    - see and use the Tools screen
    - right-click screen items and modify their settings.
    - drag-and-drop files onto the Forms screen and Other screen.
    - import settings from Motorola WaveSoft-Link or Motorola TX by deleting TxMessenger.ini.
9. Copy TxMessenger.ini from the configured computer to Disk3 (or the folder named Disk3) on the installation media. (See Section 5.2.7 for instructions on creating TxMessengerBackup.ini, which you may also want to include on this master disk.)
10. Copy the contents of the Forms, Pictures, and Sounds folders (and Messages and Demo folders if desired) from the configured computer to the corresponding folders with the same name on Disk3 (or the folder named Disk3) on the installation media. For example, copy the contents of the `C:\Program Files\Motorola TxMessenger\Forms\` folder on the configured computer to `A:\Forms\` on floppy disk 3.
11. Using the modified installer disks, install TxMessenger on a new computer and test that the configurations and files are identical to the configured computer.
12. Use the modified installer disks (“master”) on the remaining computers in the fleet. To install different sets of forms or settings on different computers, create multiple sets of master installer disks.

5 Files Installed



Other than fonts, all files are stored in TxMessenger's folder. If desired, multiple copies or versions of TxMessenger can be installed, each in its own folder.

5.1 TxMessenger.exe

`C:\Program Files\Motorola TxMessenger\TxMessenger.exe`

TxMessenger.exe is the messaging application itself. Double-click its icon to run TxMessenger.

TxMessenger.exe includes all required libraries, pictures, and sounds. It is impossible to mix up different pieces from different versions (because there's only one piece). When run, TxMessenger.exe will create any necessary folders if they are missing, and upon exiting, will create the TxMessenger.ini file if it is missing.

TxMessenger.exe includes a built-in communications engine. This means that if TxMessenger is exited or for whatever reason isn't running, messages can't be sent or received*, including high priority or emergency messages. For MW-520 users, this means the emergency button won't send an emergency message.

Important: Always leave TxMessenger running to send and receive messages!

*The radio modem can be configured in RSS to store one or more messages when the computer isn't ready or when no messaging application is running. The host receives a positive acknowledgement that the modem successfully received the message, even though the user hasn't read it.

5.2 TxMessenger.ini

```
C:\Program Files\Motorola TxMessenger\TxMessenger.ini
```

TxMessenger.ini contains modified or customized settings for TxMessenger. TxMessenger.ini should never be protected against read/write access from TxMessenger.

Beginning in v1.2, the existing TxMessenger.ini file is backed up and renamed with a numeric filename extension (like TxMessenger.001) when the installer is told to overwrite the file during installation.

The value of settings (everything after the equals '=' sign) is generally limited to 255 characters, except for scripts which are limited to 2048 characters.

5.2.1 Adding Settings

If a setting isn't specified in the TxMessenger.ini file, TxMessenger uses a built-in default value. This reduces the number of settings that must be contained in TxMessenger.ini to only those that a particular customer has altered and a few settings that reflect the last application state, such as window position and the last status message sent.

Because unmodified settings aren't listed in TxMessenger.ini, it is necessary to add missing settings, each on an individual line, by typing them in manually. It isn't necessary to add new settings in alphabetical order – simply type the setting at the end of TxMessenger.ini and it will be sorted the next time TxMessenger exits. A complete list of TxMessenger settings can be found in Motorola TxMessenger Settings.xls.

5.2.2 How to Edit TxMessenger.ini

TxMessenger.ini can be edited using the Windows accessory Notepad or any plain text editor.

1. Exit TxMessenger.
2. If the operating system is configured to associate the filename extension ".ini" with Notepad, simply double-click the TxMessenger.ini icon to open it. Otherwise, choose Notepad from Start→Programs→Accessories→Notepad, and choose Open from the Notepad File menu and select the TxMessenger.ini file.
3. Make any changes or additions desired.
4. Choose Save from the Notepad File menu.

Don't edit TxMessenger.ini in Microsoft Word or WordPad, because those applications save in a file format that TxMessenger can't read.

5.2.3 Automatic Setting Sorting

Upon exiting, TxMessenger writes all modified settings to TxMessenger.ini in alphabetical order. This makes it easy to find a modified setting and it automatically groups similarly-named settings together.

Because of sorting, all blank lines and comments on individual lines are lost. If a comment is absolutely necessary, start it with a phony setting name, like:

```
LogLevelComment=// Here is a remark
```

 **Warning:** Never mark the TxMessenger.ini as read-only. Important information cannot be saved if the TxMessenger.ini is read-only, which may result in unexpected behavior.

5.2.4 Settings File Recreated Upon Exit

Upon exiting, TxMessenger.ini is created if missing, and deleted and recreated if it already exists. Since the old TxMessenger.ini is always deleted and rewritten, the chances are minimal of a corrupt settings file causing problems.

 **Warning:** Never modify TxMessenger.ini while TxMessenger is running. Since the file is deleted and rewritten when TxMessenger exits, all changes will be lost.

5.2.5 Changing a Setting While TxMessenger Is Running

The following method will modify most settings while TxMessenger is running. However, some settings may not take effect immediately or may not take effect until TxMessenger is exited and run again.

1. Type Ctrl-t while TxMessenger is running.
2. The Tools screen appears. Click the Script Tester button.
3. The Script Tester window appears.

4. Type the setting name and value as though it were being typed into the TxMessenger.ini file, except put quotes around the value.

   `StatusBarHeight=0` would be typed as `StatusBarHeight="0"`

   `KeysClick=TRUE` would be typed as `KeysClick="TRUE"`

   `CustomerInfo=Licensed to Tom Gavin` would be typed as `CustomerInfo="Licensed to Tom Gavin"`

   If quotes are necessary within the setting value, use a backslash before each internal quote:

   `CustomerInfo=Licensed to Tom "Dangerman" Gavin` would be typed as `CustomerInfo="Licensed to Tom \"Dangerman\" Gavin"`

   If backslashes are necessary within the setting value, use a backslash before each internal backslash:

   `CustomerInfo=Licensed to Tom "Backslash (\)" Gavin` would be typed as `CustomerInfo="Licensed to Tom \"Backslash (\\)\" Gavin"`

   Quotes are necessary because, in a script, a setting can be set to the value of another setting. The quotes indicate that the value should be set exactly as typed within the quotes. Backslashes are necessary to indicate that the next character is included in the text and that the next character is not the end of the quote. But, I seriously digress.

5. Press the enter or return key.
6. If the setting change contained an error, a message appears in the Script Tester window. If the setting change was accepted, no message appears.
7. To modify a statement entered into the Script Tester window since it was last opened, press the down arrow or click the drop-down menu arrow in the upper-right corner of the Script Tester window.

5.2.6 Display All Current Settings

To view a list of all settings (including defaults) and their current values, type ctrl-t to display the Tools screen and click the Display Current Settings button.

The Tools screen wasn't available in version 1.2 or earlier. For those versions of TxMessenger, type SHOW "SETTINGS" into the Script Tester window and press return. Individuals that need more convenient access, such as administrators or technicians, can create a function key or Other button to show the settings list.

The settings list is displayed on the Desk in TxMessenger. Performance may decrease while the settings list is active, since it is rebuilt and redrawn every time a setting changes. Clear the screen, display a form, or display a message to deactivate the settings list display.

5.2.6.1 Icons in List of All Settings

The setting value is encrypted in the TxMessenger.ini file. See PUBLIC and PRIVATE script commands.

The setting value has been modified or was read from TxMessenger.ini.

The setting value is locked and can't be changed.

The setting value won't be written to the TxMessenger.ini file, even if it is modified.

The setting value is the internal default value and wasn't read from TxMessenger.ini.

5.2.7 Corruption-Recovery Backup Settings File

[Not available in versions prior to v3.5]

At startup, if the TxMessenger.ini is missing or does not contain any valid settings, TxMessenger will attempt to read the settings from the TxMessengerBackup.ini (if the file exists).

TxMessenger doesn't create or update this backup file; it must be created and updated manually by the administrator as desired. Here is the suggested usage:

1. Fully install, configure, and test TxMessenger.
2. Copy the TxMessenger.ini file and name the copy TxMessengerBackup.ini
3. Perhaps set TxMessengerBackup.ini to read-only or use other operating system protections to prevent ordinary users from modifying the backup file.

If something then happens to TxMessenger.ini, such as the file is deleted or emptied, TxMessenger automatically restores the settings from TxMessengerBackup.ini (which is the way they were in step 1 above). Alternatively, if TxMessenger.ini is still readable but partially corrupted (or if for whatever reason the administrator wants the tested configuration restored), the user can be instructed to quit TxMessenger, delete TxMessenger.ini, and run TxMessenger. At that point, TxMessenger automatically restores the settings from TxMessengerBackup.ini.

 **Warning:** This feature came about because some customers using Microsoft Windows XP were powering off their computers without first properly shutting down Windows. Files were not being allowed to be correctly written to the disk, such as TxMessenger.ini (resulting in a loss of configuration and license information). Regardless of the application, improperly shutting down Windows can result in file corruption and other problems, and should be strictly avoided

5.3 Forms folder

`C:\Program Files\Motorola TxMessenger\Forms\`

The Forms folder contains all of the forms that a user can access in TxMessenger. These forms are displayed in the Forms screen, when the host transmits data to merge with the form, and when a function key (or other script) shows a form.

All subfolders (and files within subfolders) within the Forms folder are ignored.

Forms downloaded from the host are saved into the Forms folder. Forms created locally are also stored in the Forms folder.

TxMessenger can read TX-protocol Motorola TX and Motorola WaveSoft-Link form file formats without any manual conversion necessary. Simply copy the *.fm, *.dfm, or *.fmt forms directly into the Forms folder. WaveSoft-Link compiled forms, *.dcf, are not supported.

Forms can be dragged from the Windows File Manager onto the Forms screen of TxMessenger to convert them to TX format and save them as text files (for easy editing in Notepad) in the Forms folder. *(This feature isn't available in versions prior to v1.2).*

The Motorola Form Converter tool can convert *.fm, *.dfm, or *.fmt forms to *.txt forms so that they can easily be modified in Notepad. Form Converter can also convert forms between the different file formats so that the forms can be edited with other tools, such as the Motorola TX Form Gen tool. Form Converter and its documentation are available on the shipped media in the Motorola Form Converter folder.

TxMessenger created or downloaded forms are stored in text file format. This makes it easy to create or modify forms using Notepad. See the Form Creation Tutorial.

Forms have internal names that uniquely identify them. The internal name is located at the 3rd and 4th byte of the TX-protocol header. Example internal names are TA, TB, DC, M1, and so on.

The TX-protocol may technically allow any two character combination for an internal name except " " (two spaces) or 0x0000 hexadecimal. But, in practice, two uppercase letters or numbers are safest and less confusing. For historical reasons and host compatibility, the most popular choices are TA through TX.

The internal name is often more important than the filename itself, so much so that most form filenames are named after their internal name, like TA.fm. Although perfectly acceptable, TxMessenger doesn't require the form filename to match the internal name.

Depending on the desires of the system administrator, it may be more useful to either make the filename match the internal name or it may be more useful to make the filename something descriptive, like Stolen Property Inquiry.txt. However, don't name a filename to something that is a legitimate internal name, such as TB.txt, if the internal name is actually something different, like TC.

If more than one form in the Forms folder contains the same internal name, only one of the forms is made available and displayed in the Forms screen. For example, if TE.fm and TE.fmt and Sample.txt (all with an internal name of TE) are stored in the forms folder, only one is displayed on the Forms screen. The choice of which form is honored is arbitrary and should not be relied upon, as it depends on the order in which the operating system reveals them to TxMessenger.

If a form with the same internal name exists in the Forms folder and is downloaded from the host, the existing form is deleted (regardless of its filename) and the downloaded form is saved. To prevent downloaded forms from overriding existing forms, set `DownloadedFormCanReplaceExisting=FALSE`. Individual forms can be protected by setting them to read-only in the Windows file properties.

The internal name of a form is used to merge downloaded or draft fields back into the form. If a form is modified, the previously saved fields may no longer fit into the form or be placed into the correct fields. If a form is deleted or the internal name changed, such as TA to TB, the saved fields won't be able to locate the form and will fail to display properly.

Plain text files or forms with an internal name of " " (two spaces) can be placed in the Forms folder to provide easy-to-access notes or instructions. The notes can be selected and displayed by the user in the Forms screen.

5.4 Pictures folder

```
C:\Program Files\Motorola TxMessenger\Pictures\
```

The Pictures folder can contain bitmaps that appear in the Forms, Coded, Status, or Other screens or within forms and messages themselves. This folder is not necessary and may be empty.

All subfolders (and files within subfolders) within the Pictures folder are ignored.

When making a change to the Pictures folder, it may be necessary to exit TxMessenger and run it again for the change to take effect.

Pictures must be in Windows bitmap file format and have the filename extension ".bmp". TxMessenger doesn't support GIF, JPEG, ICON file formats, OS/2 bitmaps, or any other picture format. A simple method of conversion is to display the desired picture using an application that supports it, take a screen shot (Alt-Print Screen key) and paste into the Windows Paint accessory. Then save the Paint window, which will automatically save the screen shot in bitmap format.

When TxMessenger displays most pictures, it first looks in the Pictures folder. If the picture isn't found, TxMessenger looks into its built-in pictures inside the TxMessenger.exe file. For that reason, most built-in pictures can be overridden with a picture of the same name in the Pictures folder.

With version 3.0 and higher, an exiting image is shown when the user exits TxMessenger. The built-in image can be overridden by placing an Exiting.bmp image in the Pictures folder.

5.4.1 List of Built-in Pictures

For a list of built-in pictures (and the picture names to override):

1. Exit TxMessenger.
2. Copy the contents of the *Copy Us All Into Forms Folder To View All Of The Built-In Pictures* folder from the Master Picture Collection folder on the shipping media. Paste the contents into the Forms folder.
3. Run TxMessenger.
4. Click the Forms button to see the Forms screen.
5. Click individual pictures in the Form screen for more information.

5.4.2 Associating a Picture with a Form

A simple way to associate a picture with a form on the Forms screen is to name a picture the same name as the form name, except for the file extension. For example: to have the FormGun.bmp picture displayed for the TA.fm form, copy the FormGun.bmp picture from the *Bitmap Files Of Built-In Pictures Individual Files Can Be Copied Into Pictures Folder And Altered Or Renamed* folder in the Master Picture Collection on the shipping media to the Pictures folder. Rename the FormGun.bmp picture to TA.bmp.

5.5 Sounds folder

```
C:\Program Files\Motorola TxMessenger\Sounds\
```

The Sounds folder can contain Wave-format sound files to play when certain events occur in TxMessenger. This folder is not necessary and may be empty.

All subfolders (and files within subfolders) within the Sounds folder are ignored.

Sounds must be in Windows Wave file format and have the filename extension ".wav". TxMessenger doesn't support any other sound formats.

When TxMessenger plays a sound, it first looks in the Sounds folder. If the sound isn't found, TxMessenger looks into its built-in sounds inside the TxMessenger.exe file. For that reason, all built-in sounds can be overridden with a sound of the same name in the Sounds folder.

Individual sound volumes can be changed by resampling a sound at a higher or lower volume level. Sound editing tools are commonly available that can resample sounds at different volumes and can also make and digitize new sounds.

Note: TxMessenger can only play one sound at a time and therefore will cut off a sound already playing in order to play a new sound. This effect is especially apparent when attempting to play a long sound.

A more subtle problem is when TxMessenger plays a short sound like a click, thus cutting off the sound preceding it. For example, if you want to play a sound when a Nak is received and then show the message screen, you would do the following:

```
NakScript = SHOW "Messages"; PLAY SOUND "ReceivedNak"
```

If you were to reverse the two commands above, the click sound that is automatically played when SHOW "Messages" is run would prevent the

ReceivedNak sound from finishing if it is too long or from playing at all. You could also insert a SLEEP command in between sounds to allow one to finish before the next one starts.

5.5.1 List of Sound Events

BeepLong.wav†
BeepShort.wav†
ButtonDown.wav†
ButtonUp.wav†
DeleteMessage.wav
EndOfField.wav†
KeyClick.wav†
Minimize.wav
ModemReceiving.wav
ModemTransmitting.wav
ReceivedAck.wav
ReceivedCallDispatch.wav
ReceivedCoded.wav
ReceivedHighMessage.wav†
ReceivedLowMessage.wav†
ReceivedNak.wav
ReceivedNormalMessage.wav†
ReceivedStatus.wav
ReceivedTime.wav
ReceivedUrgentMessage.wav†
RepeatingUnreadMessage.wav†
SelectionMoved.wav†
Shutdown.wav
Startup.wav†

(†Built-in)

5.5.2 Disabling a Specific Sound

1. In the Windows Notepad accessory, create a new, blank document by choosing New from the File menu.
2. Save the blank document into the Sounds folder with the same name and filename extension as the sound to disable. For example, to turn off the sound that TxMessenger makes at startup, save a blank text file named "Startup.wav" in the Sounds folder.

5.5.3 Playing a Sound Not Listed as an Event

Any sound, either built-in or totally new, can be played whenever a script is executed, such as on function keypress. See `Play Sound` in the Script Command Reference.

5.5.4 Disabling Sounds

Immediately before a sound is played, TxMessenger checks the setting `SoundsEnabled`. If `FALSE`, no sounds or beeps are played.

5.5.5 Speaker Beeps Instead of Sounds

Immediately before a sound is played, TxMessenger checks the setting `SoundsBeep`. If `TRUE`, the default Windows sound is played instead. If Windows can't play the default sound (for example, on computers without a sound card), the speaker is beeped instead.

5.5.6 Playing a Beep Instead of a Specific Sound

1. In the Windows Notepad accessory, create a new, blank document by choosing New from the File menu.
2. Type `Beep`. (Must read "Beep" exactly. No extra spaces, carriage returns, punctuation, or any extra text.)
3. Save the document into the Sounds folder with the same name and filename extension as the sound to replace with a beep. For example, to turn off the sound that TxMessenger makes at startup, save the text file with the name "Startup.wav" in the Sounds folder.

5.6 Messages folder

```
C:\Program Files\Motorola TxMessenger\Messages\
```

The Messages folder contains all the messages in TxMessenger, including received, saved, sending, sent, and trash. TxMessenger must have adequate disk space and read/write access to the Messages folder. Otherwise, messages may be lost.

All subfolders (and files within subfolders) within the Messages folder are ignored.

Don’t edit or manipulate messages except using TxMessenger. Deleting message files while TxMessenger is running will result in message loss. Altering, adding, moving, changing the properties, or renaming message files at any time will result in message loss.

Deleting any or all message files when TxMessenger is not running is a permissible method of deleting messages.

Don’t copy Motorola TX, Motorola WaveSoft-Link, or any other file formats into the Messages folder. TxMessenger only supports the TxMessenger message file format.

5.7 Uninst.isu

`C:\Program Files\Motorola TxMessenger\Uninst.isu`

The Uninst.isu file contains information about the files and folders created by the TxMessenger installer. The uninstaller may not work if this file is removed, altered, or renamed.

Note: This file will not be installed when installing TxMessenger v1.2 or greater. If it exists from previous installations, it may be removed.

It isn't necessary to uninstall TxMessenger before reinstalling or installing a newer version.

5.7.1 To Uninstall TxMessenger:

1. Choose Start→Settings→Control Panel.
2. In the Control Panel folder, choose Add/Remove Programs.
3. In the Add/Remove Programs window, choose Motorola TxMessenger from the list.
4. Click the Add/Remove button.
5. Follow any remaining prompts.

5.8 Scripts Folder

`C:\Program Files\Motorola TxMessenger\Scripts\`

The Scripts folder is a useful place for storing your custom scripts. This keeps the TxMessenger folder organized if there are a large number of custom scripts. The Scripts folder does not contain any scripts by default. TxMessenger does not specifically look for scripts in the folder.

All subfolders (and files within subfolders) within the Scripts folder are ignored.

Note: This folder is not created in versions prior to v2.5.

6 Troubleshooting

6.1 Technical Support Information

If there's a problem that can't be resolved after reading this document and the release notes, gather the following pieces of important information before contacting technical support.

6.1.1 Version number

TxMessenger's version number appears in the Windows file properties and splash screen (see sections below).

- Development versions contain the letter 'd'. Motorola internal use – completely untested. Example: 1.0d9
- Alpha versions contain the letter 'a'. Motorola supervised external use – not lab tested. Example: 1.0a14
- Beta versions contain the letter 'b'. External use – expected to be stable but not fully certified. Example: 1.0b15
- General release versions don't contain a letter. Example: 1.0

Only general release versions should be in use by actual users. All other versions (development, alpha, and beta) are uncertified test versions and should not be relied upon. Test versions may be provided to a site to determine if a defect correction or new feature is working adequately. Test versions should be discarded and replaced with the next general release as soon as possible.

The letters after a test version number indicate which next general release this version tests. For example: 1.0a14 was the fourteenth test of version 1.0. Because the test versions come prior to the general release, 1.0a14 is older (and less robust) than is 1.0. The next test version after 1.0 would be 1.1d1 (or 1.1a1 if released externally).

From oldest to newest, version history might go something like this: 1.0d1, 1.0d2, 1.0d3, 1.0a5, 1.0d6, 1.0d7, 1.0d8, 1.0a10, 1.0a11, 1.0a12, 1.0d13, 1.0a13, 1.0a14, 1.0a15, 1.0a16, 1.0 (general release), 1.1a1, 1.1d2, 1.1a2, 1.1b3, 1.1

6.1.2 Information in Windows

Right-click TxMessenger.exe and choose Properties to display TxMessenger's:

- Location (also known as the path or the directory).
- File size, in both MB (megabytes) and exact bytes.
- Creation date and time.
- Modified date and time.

Click the Version tab to display:

- TxMessenger's version number.

From the Windows task bar, choose Start→Settings→Control Panel. In the Control Panel, choose System to display:

- The operating system name, version, and service pack (if installed).
- The computer manufacturer and model name (if provided by the manufacturer).
- Customer specific information (if provided).

6.1.3 Information on Physical Devices

- The computer manufacturer name, model, revision number (if any) and serial number.
- The external radio modem manufacturer name, model, revision number (if any) and serial number.

6.1.4 Information in About TxMessenger

While TxMessenger is running, click the About TxMessenger button located in the Other screen. This screen is the same as the splash /startup screen and describes:

- TxMessenger's version number.
- The operating system name, version, and service pack (if installed).
- The computer manufacturer and model name (if provided by the manufacturer).
- Customer specific information (provided by the `LicenseCustomerName` setting in the TxMessenger.ini).
- Whether or not GPS is enabled. The GPS enabled icon is displayed in the lower-right hand corner or on the right side of the TxMessenger splash screen if `GPSLicenseNumber` is set to a valid GPS license number (not available in versions prior to v2.0):

- With version 3.5 and higher, a service status icon has been added to the About screen. This icon will show one of five scenarios:

    1. Whether a session is available and enabled.
    2. Whether encryption* is available and enabled
    3. Whether compression* is available and enabled
    4. Whether all three are available and enabled
    5. Whether none are available or enabled

*A session must be available and enabled before encryption and/or compression can be available even if they have been enabled.

6.1.5 Information in Modem Status

While TxMessenger is running, click the Modem Status button located in the Tools screen (in versions prior to v1.3, the button is located in the Other screen). The modem status screen describes many communication conditions. A screen shot (Alt-Print Screen) or description should be created to help diagnosis possible problems.

To aid troubleshooting, the modem status screen can be displayed in a separate window from TxMessenger by setting `ModemStatusWindowIsSeparate=TRUE`.

The modem status screen describes:

- Modem:
  - Online (channel acquired and link established to the network to send and receive SDU packets – good)
  - MWCS (TxMessenger interfaces to the modem through MWCS and not directly through a COM port)
  - Offline (modem in local command mode, not ready to send nor receive SDU packets)
  - Not detected (`ModemPort` or `ModemBaud` may be incorrectly set)
  - Startup Failed (modem startup / initialization routine failed)
  - Not Used (enabled by `ModemUsed=FALSE`), or
  - Demo (enabled by `ModemUsed=FALSE` and a folder named Demo exists in the same folder as the application)

- Host:
  - Up (good)
  - Down (host problem), or
  - Unknown

- Range:
  - In Range (good)
  - Out of Range / Scanning (physically too far away from an RF antenna, a base station or antenna problem exists, the host is in the dekeyed portion of its cycle, or the modem is not registered),
  - Voice, or
  - Unknown

- Modem Registered (sometimes shortened to "Modem Regi…"):
  - Registered
  - Not Registered (modem ID needs to be registered on the RNC or WNG box), or
  - Unknown

- Modem ID: ######## (unique modem identifier in hexadecimal notation)

- Acks: Number of inbound messages (sent to the host) that have been acknowledged by the Radio Network Controller.

- Naks: Number of inbound message that have been not acknowledged by the Radio Network Controller and ultimately not delivered to the host.

- Bytes In: Number of bytes read from the modem.

- Bytes Out: Number of bytes written to the modem.

- Rx Quality:
  - less than 0x40 (good),
  - 0x40 to 0xBF (moderate),
  - 0xCO to 0xFF or blank (physically out of range, base station problem, antenna problem on modem, wrong antenna in use (primary vs. secondary)).

Unlike RSS, the modem Rx Quality value only updates when the modem chooses to supply this information. The value is not regularly polled from the modem.

- Com Port: Usually 3 for internal modems and 1 for external modems. Make sure this matches the port to which the radio modem is connected.

- Baud Rate: The baud rate between the DTE (e.g. laptop, MW-520) and the DCE (i.e. radio modem). This is not the data throughput over the air.

- Flow Control:
  - RTS/CTS (preferred)
  - XON/XOFF
  - None (not recommended)

- Connected:
  - Connected and ON
  - Disconnected or OFF
  - Unknown

- Frequency: Displays the radio modem's transmission frequency as well as a 2 byte channel number in parentheses (which contains the band and the radio channel – channel number not shown when MWCS is enabled).

- Antenna:
  - Automatic (modem determines which antenna to use)
  - Primary (only the primary antenna is used)
  - Secondary (only the secondary antenna is used)
  - Unknown

- Rx Status:
  - Enabled (good)
  - Disabled (the modem is prevented from receiving messages)
  - Unknown

- Tx Status:
  - Enabled (good)
  - Disabled (the modem is prevented from transmitting messages)
  - Unknown

- Modem S/W: the radio modem hardware's firmware version number.

- Modem H/W: the radio modem hardware model number (or sometimes the manufacturing date)

- Last Error: blank (good), otherwise a text description of the last error encountered.

6.2 Files to Provide for Technical Support

Log files are necessary to diagnose most issues. In addition, technical support is greatly improved by providing copies of the TxMessenger.ini file, forms, messages, pictures, and sounds. See Backing Up for a list of files and directories necessary to recreate a mobile device environment.

6.3 Enabling Logging

TxMessenger can produce two different logs. Both logs:

- Are located in the same folder as the TxMessenger.exe application.
- Can be viewed using the Windows Notepad or WordPad accessories.
- Are limited to the size specified by the `LogSizeBytesMaximum` setting.
- Can be turned completely off by setting their severity level to `0`.
- Report only the most severe events at severity level `1` (default).
- Report the maximum number of events (including notes and least severe events) at severity level `10`.
- In version 3.0 and higher, specific log levels can be disabled. For instance, if `LogLevel` is set to 10, log level 5 can be disabled if `LogLevel5Disabled` is set to TRUE. Or if `ModemLogLevel` is set to 6, modem log level 3 can be disabled if `ModemLogLevel3Disabled` is set to TRUE.

TxMessenger.log reports application-related events. The severity level is controlled by the `LogLevel` setting.

MCM.log reports communication-related events. The severity level is controlled by the `ModemLogLevel` setting.

6.3.1 Recording the Most Events in Both Log Files

1. Exit TxMessenger.
2. Delete `C:\Program Files\Motorola TxMessenger\TxMessenger.log` and `C:\Program Files\Motorola TxMessenger\MCM.log`.
3. Open `C:\Program Files\Motorola TxMessenger\TxMessenger.ini` in Notepad.
4. Set `LogLevel=10`
5. Set `ModemLogLevel=10`
6. Save TxMessenger.ini.
7. Run TxMessenger.
8. Perform whatever actions are necessary to reproduce the problem being reported.
9. Exit TxMessenger.
10. Edit TxMessenger.ini to restore `LogLevel=1` and `ModemLogLevel=1` (Do not forget this step. Performance and disk space decreases when logging is fully active.)

6.3.2 Activating Logging While TxMessenger Is Running

Logging can be turned on briefly during a particular time period by programming a function key or running a script immediately (Ctrl-t when TxMessenger is running and then click the Script Tester button).

```
LogLevel=10;ModemLogLevel=10; /* Script to turn logging to
maximum */

LogLevel=1;ModemLogLevel=1; /* Script to restore normal
logging. */
```

6.4 Testing a Script

If a configured button or script doesn't perform as expected, it is easy to test it and receive feedback in the Script Tester window.

6.4.1 Script Tester Window

1. Type Ctrl-t while TxMessenger is running.
2. The Tools screen appears. Click the Script Tester button.
3. The Script Tester window appears.
4. Type (or copy and paste) the script to be tested into the field at the top of the Script Tester window.
5. Press return or enter on the keyboard.
6. The results of the script and any errors are displayed in the bottom portion of the Script Tester window. Nothing is displayed if there aren't any results or errors.
7. To modify a statement entered into the Script Tester window since it was last opened, press the down arrow or click the drop-down menu arrow in the upper-right corner of the Script Tester window.

6.4.2 Getting More Responses in the Script Tester Window

To display the value of a setting or to determine what line is executing, the RESPOND command can be used to display text in the Script Tester window.

For example, typing this script into the field in the Script Tester window (it must be crammed onto a single line):

```
temporary=TRUE;
RESPOND temporary;
RESPOND "\x0D\x0A"; /* move down a line */
IF temporary=FALSE;
     RESPOND "Now I'm here.";
ELSE;
     RESPOND "Must have been true.";
ENDIF;
WHOOPS /* This line generates an error */
```

The Script Tester window displays:

```
TRUE
Must have been true.
#11101: No such command
```

6.4.3 Storing a Result in a New Setting

Sometimes it is helpful to store data in a setting and review it after the script has completed or TxMessenger exits.

```
SAVE;
/* Remember current message on desk, if any */
temporary=DeskMessageID;

NEW MESSAGE; /* Clear the screen */

/* Set the screen to “This works” */
SET FIELD 1 TO "This works";

/* Set LaterAlligator setting to “This works” */
SET LaterAlligator TO FIELD 1;

CLEAR; CLEAR MORE; /* Clear the screen without saving
the current message (“This works”) to the message list
*/

SHOW MESSAGE temporary; /* Display the original message,
if any, on the desk */

EXIT; /* What a waste. All that work to restore the
message and you only wanted to use it as an example. */
```

TxMessenger.ini now contains `LaterAlligator=This works`. Of course, a simpler example would be:

```
LaterAlligator="Why didn’t you just say so."
```

To remove a setting from TxMessenger.ini via a script:

```
DELETE LaterAlligator
```

7 Script Command Reference

7.1 Parameters

"*boxName*" — A text string that uniquely identifies a box (such as `"Forms"`). Not case sensitive, but spaces and all other characters are significant. The name is truncated to a maximum of 255 characters, or is truncated at character 0 (end-of-string) if character 0 is imbedded. Box names are: about, coded, command, desk, extras *(extras not available in versions prior to v1.5)*, forms, keys, log, messages, modem, modem log, other, script tester, settings, status, tray *(tray not available in versions prior to v3.0)* and tools *(tools not available in versions prior to v1.3)*. Not all boxes are appropriate in all contexts.

"*fieldName*" — A number or text string that uniquely identifies a form field. Not case sensitive, but spaces and all other characters are significant. The name is truncated to a maximum of 255 characters, or is truncated at character 0 (end-of-string) if character 0 is imbedded.

If a number is used and no field is named that number, the number specifies the nth field in the form. (1 is the first field from the top, 2 is the second field, and so on.) When a text message or clear screen is present on the desk (instead of a form), the entire contents of the desk are considered field 1.

The content of password fields is accessible through the scripting language.

"*formName*" — A text string that uniquely identifies a form. Not case sensitive, but spaces and all other characters are significant. The name is truncated to a maximum of 255 characters, or is truncated at character 0 (end-of-string) if character 0 is imbedded. If the form name is exactly two characters in length and matches the TX internal form name of a currently installed or downloaded form, the form name will be accepted as representing a TX internal form name. Otherwise, the form name will be assumed to be the filename.

number — Base 10 is assumed unless beginning with `0x` or `0X` in which case base 16 (hexadecimal) is indicated, or unless beginning with `0b` or `0B` in which case base 2 (binary) is indicated. Unary negative (‘`-`’) and unary positive (‘`+`’) accepted. Decimal points (‘`.`’) accepted with or without a leading `0`, even if the number is ultimately cast as an integer. No expressions or scientific notation allowed. Although not preferable, numbers can be specified with quotes.

"*text*" — A text string. Case sensitive. Quotes required (simple quotes `""`, not smart quotes “”). Within the quotes, use `\n` to imbed a linefeed, `\t` for tab, `\`*nnn* for a decimal byte, `\x`*nn* or `\X`*nn* for a hexadecimal byte, `\"` for a quote, `\\` for a backslash. The text string is truncated at character 0 (end-of-string) if character 0 is imbedded, but the script execution continues. Numbers without quotes can be used whenever a text string parameter is specified, such as `-45` instead of `"-45"`.

settingName

A character string that uniquely identifies a setting. Not case sensitive. Alphanumeric and '_' only. Must not begin with a numeric. Truncated (significant) to maximum of 255 characters, but the script execution continues.

Certain words are reserved words in certain contexts (FORM, MESSAGE, FIELD, NEXT, PREVIOUS, CODED, EMERGENCY, STATUS, DESK, etc.) and may not be used as a *settingName* in certain commands (so, preferably not at all).

Some settings are predefined constants, such as `TRUE` and `FALSE`.

A *settingName* may be substituted in the place of any other parameter.

`temporary` is the name of a setting used by internal scripts. You may also use this setting, but its contents cannot be relied upon after a particular script ends or another script is called (which can happen in the middle of your script if you change a setting, call a script, or execute a command that causes a script to be called). `temporary` is never saved to TxMessenger.ini.

@ Substitutions:
Beginning in TxMessenger v3.0, setting names may include one or more `@` characters. Each `@` is replaced with the contents of the `substitute` setting at the moment that *settingName* is executed.

```
substitute="o";
Motorola="Hello";
RESPOND M@t@r@la;
```

The script above outputs the word "Hello".

The real purpose of this substitution feature is to allow a script to programmatically access a numbered setting. For example, normally when a button on the Tray panel is clicked, the setting `ButtonTrayPanelSelected` is set to the button's number. With the @ character, a script can now examine a specific setting name on the fly, based on the selected button.

```
ButtonTrayPanelScript=
      substitute=ButtonTrayPanelSelected;
            // Let's say ButtonTrayPanelSelected is 5
      ALERT ButtonTray@Position;
            //  This would become ButtonTray5Position
```

The script above causes the position of the button being clicked to be displayed in an alert box. It does this by getting the number of the button and storing that in the substitute setting. Let's say 5. Then when the ALERT command is executed, the value of the setting name ButtonTray5Position is read. Of course the script could examine a label or color or font or whatever with an `IF` and can even change it.

If the substitute setting changes and that line is rerun (like with a `GOTO`), the resulting setting name may be different each time that portion of the script is interpreted.

 Substitution Warnings:

1. The @ character substitution is only for setting names (which also can be used in the place of text, number, boxName, fieldNames, and so on). The @ character can't be used elsewhere, such as within command words.

```
substitute="ALERT";
@ "HELLO!";
```

The above script is invalid, since @ characters aren't allowed in a command word.

2. It is possible to create invalid setting names if you aren't careful. Although the results are usually benign, the results aren't defined or supported. For example:

```
substitute="    ";
spaces@arent@allowed=1;
```

The above script is invalid, since spaces aren't allowed in setting names.

Please stick to using the @ character to create valid *settingNames*.

3. Be aware that other scripts may execute during your script, such as a result of a command, calling another script, or changing a setting. If substitute is modified by the other script, you may not get the results you expected:

```
substitute="1234";
ButtonTray@Label="Hello";
      // ButtonTray1234Label="Hello"
RUN FILE "OtherScript.txt";
      // Let's say the other script changed
      // the value of substitute to "Grandma"
ALERT ButtonTray@Label;
      // Then this won't necessarily say "Hello"
      // Because it now works out to be:
      // ButtonTrayGrandmaLabel
```

The above script is valid, but unreliable.

To avoid such problems, it is best to set the value of substitute to the desired value immediately before using a *settingName* that contains an @.

7.2 Separators

<*eos*>	An end-of-string (0x00) ends a script. This character is required at the end of scripts passed in through the API, but isn't necessary at the end of function key scripts or scripts received from the host.
<*end-of-line*> <*carriage return*>	Considered whitespace. A command may span multiple lines. In other words, `NEW` `MESSAGE` is the same as `NEW MESSAGE`.
32 <= ASCII	All unquoted characters less than or equal to 32 decimal (except 0), including end-of-lines or carriage returns, are considered whitespace.
whitespace	Multiple, leading, or trailing whitespaces are ignored.
`""`	Whitespace is not required before a leading quote or after a trailing quote. A trailing quote must be included before the end of script or none of the script is executed. Always use simple quotes `""`, not smart quotes “”. If any of the script contains an unterminated quote, none of the commands will be executed.
`;`	separates multiple commands. Multiple, leading, or trailing semicolons are ignored.
`//`	End of command line comment. All characters following the `//`are ignored until the end of the line. Therefore, `NEW//hello` `MESSAGE` is the same as `NEW MESSAGE`.
`/* */`	Block comment and its enclosed characters are ignored. The block comment must be closed before the end of the script or none of the script is executed. Block comments can't be nested. Block comments are considered whitespace, so `NEW/*hi*/MESSAGE` is the same as `NEW MESSAGE`. If any of the script contains an unterminated block comment, none of the commands will be executed.
label`:`	A character string that uniquely identifies the start of a command. Not case sensitive. Alphanumeric and '_' only. Must not begin with a numeric. Truncated to maximum of 255 characters, but the script execution continues. Must appear before the start of a command; cannot appear within or at the end of a command. Multiple labels for the same command are acceptable. If any of the script contains an invalid, blank, or duplicate labeled command, none of the commands will be executed.

7.3 Commands

ADD *settingName* BY *number*	Converts setting to a number, adds number to it, and stores the result back into setting.
ALERT "*text*"	Displays a warning dialog with the specified text.  The ALERT dialog is modal. Normal functions, including communication may not be possible while the dialog remains open.
APPEND *settingName* WITH "*text*"	*[Not available in versions prior to v1.2]* Connects the specified text to the end of the current value of the setting and stores the value back into the setting.
ASK "*text*"	*[The ASK command is not available in versions prior to v2.5]* Displays a dialog box with the text specified by the first parameter. The dialog won't show if there isn't some method for the user to dismiss it. For instance, if AskDismissWithKey and AskDismissWithClick are both FALSE.
ASK "*text*" "*text*"	Same as above, but with additional text for the label of a button.
ASK "*text*" "*text*" "*text*"	Same as above, but with two buttons.
ASK "*text*" "*text*" "*text*" "*text*"	Same as above, but with three buttons. When ASK is invoked, the dialog is displayed and the script stops until the dialog is dismissed. Numerous settings are available that ASK uses to determine colors, pictures, and methods of dismissing the dialog. It is probably best to set **all** of the settings the way you want before calling ASK, just in case the previous caller messed up the settings. After the dialog is dismissed, the name of the button label or reason it was dismissed (KEY 0x## c, CLICK, TIMER, or SCRIPT) is stored in the setting, AskNoteAnswer.  The ASK dialog is modal. Normal functions, including communication may not be possible while the dialog remains open.

`CLEAR`	Clears all of the fields of the current form. Doesn't affect messages and won't clear any read-only portion of a form.
`CLEAR MORE`	Clears the current message or clears all of the fields of the current form. If the fields are already clear, this will clear the entire form. (For backwards compatibility with WaveSoft-Link.)
`CLEAR TEXT`	*[Not available in versions prior to v1.6]* Clears the selected text. Read-only text is unaffected.
`CONVERT` *settingName* `TO BOOLEAN`	Interprets the current value of setting as Boolean (true/false) and stores the interpreted value back into setting.
`CONVERT` *settingName* `TO CHARACTER`	Interprets the current value of setting as an integer and stores that back as a single byte / character. Opposite of CONVERT TO VALUE.
`CONVERT` *settingName* `TO INTEGER`	Interprets the current value of setting as an integer (number without decimal places) and stores the interpreted value back into setting.
`CONVERT` *settingName* `TO NUMBER`	Interprets the current value of setting as a floating-point number (has decimal places) and stores the interpreted value back into setting.
`CONVERT` *settingName* `TO VALUE`	Interprets the byte value of the first character of setting and stores that back as an integer. Opposite of CONVERT TO CHARACTER.
`COPY`	*[Not available in versions prior to v1.6]* Copies the selected text to the clipboard. Does nothing if no text is selected.
`COPY "`*text*`"`	*[Not available in versions prior to v1.6]* Copies the quoted text (or contents of a setting) to the clipboard. Clears the clipboard if empty.

`DEFAULT FIELDS`	Sets all the fields of the current form to their defaults.
`DEFAULT SETTINGS`	Deletes all settings that aren't internally defined by the application. Sets all remaining settings to their application defaults.
`DELETE`	Deletes the currently displayed message if the desktop is displayed, or deletes the currently selected messages if the message list is displayed.
`DELETE ALL FORMS`	*[Not available in versions prior to v1.2]* Deletes all forms in the Forms folder and rebuilds the forms list. Files marked in Windows as read-only won't be deleted – which can be useful to protect "resident" forms.
`DELETE ALL MESSAGES`	*[Not available in versions prior to v1.2]* Deletes all files in the Messages folder and rebuilds the messages list. Files marked in Windows as read-only won't be deleted.
`DELETE FILE` *"text"*	*[Not available in versions prior to v2.5]* Deletes the file specified by text (with or without a path). Files marked in Windows as read-only won't be deleted.  Use this command with care, since it obviously provides the ability to delete important files.
`DELETE FORM "`*formName*`"`	Deletes the form specified.
`DELETE MESSAGE` *number*	Deletes the message with the ID specified.
`DELETE` *settingName*	Deletes the setting with the name specified.
`DIVIDE` *settingName* `BY` *number*	If number is not zero, then this converts setting to a number, divides it by number, and stores the result back into setting.

`DONT WRITE` *settingName*	Prevents the setting from being written to TxMessenger.ini. Delete the setting and set it to something to make it writable again.
`DRAFT DESK`	*[Not available in versions prior to v2.5]*. If a message or form w/data, either newly received or displayed from the message list, is shown on the desk, this command causes the message to be unlinked from the message list. If the user then modifies the desk or executes the SAVE command, the contents of the desk will be saved as a draft in the message list. If the original message was already in the message list, it remains unchanged.
`ELSE`	If the matching `IF` command condition evaluated to false, the script execution always continues at the `ELSE` command. (See `IF`.)
`ELSE IF "`*text*`"`…	If the `ELSE` command contains an `IF` command (without a semicolon ';' separating them), then the `IF` is evaluated to see if the command block within the `ELSE` should be executed, or whether the script should move to the `ENDIF` or to the next `ELSE`.
`EMPTY TRASH`	Permanently deletes all messages that were marked for deletion (in the trash).
`ENDIF`	Required to identify the end of an `IF` command block or `ELSE` command block. (See `IF`.)
`EXIT`	Quits the application but asks the user any final questions. (As if the user had exited from the File menu.) Terminates continued execution of the script.
`EXIT IMMEDIATELY`	Quits the application without asking the user any questions. (As if the user wasn't there.) Terminates continued execution of the script. . Exits safely even if the setting `EXIT=NEVER`.
`EXTENSION "`*text*`" "`*text*`"`	*[Not available in versions prior to v1.5]* Sends the text specified in the second parameter to the external application or device specified by name in the first parameter.
`FRONT WINDOW`	Forces the application window to the front. Depending on the operating system, this command also seems to deactivate the screen saver. However,

	the terminal remains locked if the screen saver is password protected.
`GOTO` *label*	Execution of the script continues at the command with the specified label. In an attempt to prevent infinite loops, an arbitrary limit is set on the quantity of gotos that will be executed.
`GPS READ` *settingName*	*[Not available in versions prior to v2.0]* Reads all data from the GPS device and stores the result in the setting specified by the first parameter. *Requires valid GPS license number to be entered.*
`GPS READ` *settingName "text"*	*[Not available in versions prior to v2.0]* Reads data from the GPS device up to and including the character specified by "text" and stores the result in the setting specified by settingName. If the character isn't found, the setting is cleared but data read from the GPS is retained for future reads. *Requires valid GPS license number to be entered.*
`GPS READ` *settingName "text" number*	*[Not available in versions prior to v2.0]* Same as above, but the third parameter specifies the maximum amount of time in milliseconds that TxMessenger can wait for the specified character from the GPS device. *Requires valid GPS license number to be entered.*
`GPS READ` *settingName "text" number "text"*	*[Not available in versions prior to v2.0]* Same as above, but the last parameter can be `TRUE` or `FALSE`. If `FALSE`, the data is read from the GPS but continues to be retained in memory for future reads. (This means that you will read the same data over and over again as long as `FALSE` is used.) If TRUE, all data copied into the setting is removed. *Requires valid GPS license number to be entered.*
`GPS WRITE` *"text"*	*[Not available in versions prior to v2.0]* Writes the text (or contents of a setting) to the GPS device. *Requires valid GPS license number to be entered.*
`HALT`	Terminates continued execution of the script, but not the application.

`IF "`*text*`" = "`*text*`"`

Begins a conditional command block. May be nested. A semicolon must separate the `IF` command and the rest of the block.

If either of the parameters are settings that don't exist, `""` will be substituted in their place rather than generating an error.

The two parameters are checked to see if they both match a strict definition of Booleans or numbers, and if so, the parameters are compared to each other as such. Otherwise, the two parameters are compared as strings, case sensitive.

`0x2 = +2.00`	Numeric comparison
`"2.0" = 2`	Numeric comparison
`" 2" != 2`	String comparison
`"2 " != 2`	String comparison
`"1.0A"!="1.0B"`	String comparison
`"Hi" != "HI"`	String comparison
`TRUE=1`	Boolean/numeric comparison (`TRUE` is evaluated as a 1)
`"  TRUE"!=1`	String comparison (because of the spaces within the quotes)

The comparison operators are:

=	for equal
!=	for does not equal
<	for less than
<=	for less than or equal
>	for greater than
>=	for greater then or equal

If the condition evaluates to true, execution continues.
If the condition evaluates to false, execution continues after the matching ELSE or ENDIF.
(continued)

IF command example:

```
SET test1 TO 1; /* Modify these values to test different branches. */
SET test2 TO 2;
SET test3 TO 3;
SET test4 TO 4;

IF test1 = test2; /* Notice the semicolon! */
        RESPOND "The first and second settings match.";
        RESPOND "Now moving beyond the endif.";
ELSE IF test1 = test3;
        RESPOND "The first and third settings match.";
        IF test1 != test4;
                RESPOND "But the first doesn't match the fourth!";
        ENDIF;
        RESPOND "Now moving beyond the endif.";
ELSE;
        RESPOND "The first doesn't match the second or third.";
ENDIF;
RESPOND "All done!";
```

`IF "`*text*`" EMPTY`	Same as IF (above), except evaluates to true if the text or setting doesn't have any contents at all (such as SET *settingName* to `""`). A space character or numeric value of 0 is NOT empty.
`IF DESK EMPTY`	Same as IF (above), except evaluates to true if the desk doesn't have anything on it (the splash screen is also considered empty). A space character or numeric value of 0 is NOT empty.
`IF` *settingName* `EXISTS`	Same as IF, except evaluates to true if the setting exists (regardless of the contents). A deleted or never-created setting will evaluate to false.
`IF "`*text*`" CONTAINS "`*text*`"`	Same as IF, except evaluates to true if the complete text in the second parameter exists anywhere in the first parameter. Not case-sensitive.
`IMPORT`	*[Not available in versions prior to v1.2]* The user is asked to specify a file. If a Motorola TX or Motorola WaveSoft-Link settings file is specified, TxMessenger reconfigures portions of the TxMessenger.ini settings file based on the file specified. If a form is specified, it is copied as a text file into the TxMessenger Forms folder, converted to standard TX-protocol format, and added to the Forms screen.
`IMPORT ALL`	*[Not available in versions prior to v1.2]* Searches default directories for Motorola TX and Motorola WaveSoft-Link configuration files and forms. If any configuration files are found, a dialog is presented to ask whether to import them.
`IMPORT "`*text*`"`	*[Not available in versions prior to v1.2]* Where text specifies a file (with or without a path). See `IMPORT`.

`LEAVINGIF`	If you're going to branch out of an IF with a GOTO, the ENDIF won't be reached. Use `LEAVINGIF` to indicate that the IF is ending prematurely.
	`index=1;`
	`LOOP: ADD index BY 1;`
	`IF index <= 10;`
	`  LEAVINGIF;`
	`  GOTO LOOP;`
	`ENDIF;`
	Because the ENDIF is not going to be encountered many times because the GOTO is jumping out of the IF, an `LEAVINGIF` is necessary to avoid an error message.
	
	If the GOTO is going to branch out of more than one IF, in other words nested Ifs, use one `LEAVINGIF` for each ENDIF that is being skipped.

`LOG "`*text*`"`	Stores the specified text in the log file. Does nothing if logging is completely turned off.
`LOG "`*text*`" WITH HEXADECIMAL`	Stores the specified text along with its hexadecimal value in the log file. Does nothing if logging is completely turned off. `WITH HEX` is accepted as an abbreviation instead of `WITH HEXADECIMAL`.
`LOG FILE` "text" "text"	*[Not available in versions prior to v3.0]* The contents of the second "text" get stored in the file specified in the first "text" with or without a path. If the file doesn't exist and no path is specified, the file will be created in the TxMessenger folder. If the path doesn't exist, TxMessenger will not create it.
`LOG FILE` "text" "text" `WITH HEXADECIMAL`	*[Not available in versions prior to v3.0]* The contents of the second "text" get stored along with its hexadecimal value in the file specified in the first "text" with or without a path. If the file doesn't exist and no path is specified, the file will be created in the TxMessenger folder. If the path doesn't exist, TxMessenger will not create it. `WITH HEX` is accepted as an abbreviation instead of `WITH HEXADECIMAL`.

LOGOFF	*[Not available in versions prior to v2.5]* Logs the current Windows user off the computer. Since this is operating system dependent, this command may not work completely, if at all. Note that this command has nothing to do with logging off the wireless network or CAD.
LOWERCASE *settingName*	*[Not available in versions prior to v2.5]* Converts all uppercase letters from 'A' to 'Z' of the current value of the setting to lowercase letters 'a' to 'z' and stores the result back into the setting.
MAXIMIZE WINDOW	Enlarges the current window to as large as possible.
MINIMIZE WINDOW	Hides the current window (to the task bar in Windows 95,98,NT).
MINIMODE "*text*"	***Not available. This is reserved for possible future use.*** The "text" is evaluated as a Boolean. If true, the window is shrunk to a reduced size to show shared screen space with other applications. If false (and the window is currently shrunk), the window is expanded to its original size.
MINIMODE TOGGLE	***Not available. This is reserved for possible future use.*** Same as above except it reduces the window if it isn't already reduced, or it expands the window if it is already reduced.
MULTIPLY *settingName* BY *number*	Converts setting to a number, multiplies it by number, and stores the result back into setting.
NEW MESSAGE	Begins a new message by moving to the desk and clearing it if necessary.
NORMALIZE WINDOW	Adjusts the current window to the last user defined (non-maximized and non-minimized) state.

NOTE "*text*"	*[The NOTE command is not available in versions prior to v2.5]* Displays a child window with the text specified. If a note is already displayed, it is replaced with the new note. NOTE "" will not display anything, and will close any currently displayed note. The dialog won't appear unless there is some way for the user to dismiss it. For instance, if NoteDismissWithKey and NoteDismissWithClick are both FALSE. After the note appears, the script continues immediately and the note hangs around until dismissed. Numerous settings are available that NOTE uses to determine colors, pictures, and methods of dismissing the dialog. It is probably best to set **all** of the settings the way you want before calling NOTE, just in case the previous caller messed up the settings. After the dialog is dismissed, the name of the button label or reason it was dismissed (KEY 0x## c, CLICK, TIMER, or SCRIPT) is stored in the setting, AskNoteAnswer.
PASTE	*[Not available in versions prior to v1.6]* Pastes the text contents of the clipboard starting at the current caret position.
PASTE "*text*"	*[Not available in versions prior to v1.6]* Pastes the quoted text (or contents of a setting) starting at the current caret position.
PLAY SOUND "*text*"	Plays the sound specified. A path or full filename is supported, but not required.
POSITION CARET FIELD "*fieldName*"	Moves the text entry point (caret) of the current form or message to the start of the field specified.
POSITION CARET *horizontal vertical*	(where horizontal and vertical are of type *number*) Moves the text entry point (caret) of the current form or message to the coordinates specified. Horizontal values greater than the maximum are wrapped around to the next line(s). If the resulting caret position is off the form or message, it is pinned to a maximum or minimum. Position 0 0 hides the caret. Position 1 0 places the caret in the default position (if any) for the form or message.

`POSITION WINDOW` *left top right bottom*	(where left, top, right, and bottom are of type *number*) Adjusts the window to the coordinates specified. If only left and top are specified, window will be moved to left and top but not resized.
`PRINT`	Prints the current form or message.
`PRINT "`*text*`"`	***Not available. This is reserved for possible future use.*** Prints the specified text.
`PRINT FORM "`*formName*`"`	***Not available. This is reserved for possible future use.*** Prints the form specified.
`PRINT MESSAGE` *number*	***Not available. This is reserved for possible future use.*** Prints the message ID specified.
`PRIVATE` *settingName*	Weakly encrypts the specified setting when it is saved to the setting file. This may be useful if password information is to be stored here to be later transferred to a password field with `SET`.
`PUBLIC` *settingName*	Makes the specified setting plainly visible in the setting file.
`READ FILE` *"text" settingName*	*[Not available in versions prior to v3.5]* Reads the contents of the file specified by the first "text" (with or without a path) into the setting specified. *Note:* This command was originally introduced in v3.0 with the settingName in quotes. That functionality is retained for backward compatibility although a lack of quotes is preferred (as is now shown in this manual).
`RECEIVE "`*text*`"`	Acts as though the text string specified was sent as a plain text message from the host.
`RECEIVE ACK` *number*	Acts as though the specified message ID was acknowledged as being received from the host. This should only be used in demo mode or testing, because the modem may still expect or receive a real ACK or NAK from the host.
`RECEIVE CALL DISPATCH`	Acts as though the call dispatch short-fixed message was sent from the host.

`RECEIVE CODED` *number*	Acts as though the coded number specified (0-255) was sent from the host.
`RECEIVE DATE "`*text*`"`	Acts as though a long-fixed date message was sent from the host. Eight hex characters should be specified “80MMDDYY” where month, day, and year should be in BCD format.
`RECEIVE FILE "`*text*`"`	Acts as though the contents of the file and path specified were sent as a variable-length message from the host. The file should begin with a header (usually “TX”, form name, routing bits, form type, 0x00).
`RECEIVE NAK` *number*	Acts as though the specified message ID was negatively acknowledged as being received from the host. This should only be used in demo mode or testing, because the modem may still expect or receive a real ACK or NAK from the host.
`RECEIVE STATUS` *number*	Acts as though the status number specified (0-255) was sent from the host.
`RECEIVE TIME "`*text*`"`	Acts as though a long-fixed time message was sent from the host. Eight hex characters should be specified “80HHMMSS” where hours, minutes, and seconds should be in BCD format.
`REFRESH WINDOW`	Redraws the main window. (Although usually a setting change causes the window to be redrawn automatically.) Note: This doesn’t reinterpret or reload a form or message. It simply redraws what is already there. So some setting changes may not take effect until the next form or message is brought to the desk, even if refresh window is used. See `RELOAD DESK`.
`RELOAD DESK`	Saves the current desk and then reopens it with current settings. If the form or message has a built-in script, it reruns. This command should be used with care, because although most desktop contents can be reloaded, some messages or forms (especially externally opened forms) may fail to reload or may be replaced with internal forms or messages of the same name. Note: This reinterprets the message with the current settings and then redraws the desk portion of the screen. To simply redraw the window, see `REFRESH WINDOW`.

`RESPOND "`*text*`"`	The text, number, or contents of the specified setting are returned to originator of the script (the API caller or host). Result is a string. String is empty if setting is not found. No cleaning or bounds checking is performed on the contents of setting. This command is very useful for displaying settings or text in the Script Tester window.
`RESTART`	*[Not available in versions prior to v2.5]* Shuts down Windows and restarts the computer. Since this is operating system and hardware dependent, this command may not work completely, if at all.
`RUN DESK`	*[Not available in versions prior to v1.6]* Runs the script (if any) built into the form or message currently displayed on the desk. In an attempt to prevent infinite loops, an arbitrary limit is set on the depth of `RUN DESK` commands that will be executed.
`RUN FILE "`*filename*`"`	*[Not available in versions prior to v1.6]* Executes a script from a file. In an attempt to prevent infinite loops, an arbitrary limit is set on the depth of `RUN FILE` commands that will be executed. *[Not available in versions prior to v3.5]* If the filename specified is not found, TxMessenger will automatically search for a built-in script with the same name.

`RUN` *`settingName`*	*[Not available in versions prior to v1.6]* Executes a script from the specified setting. This is a great way to create subroutines. For example:
	`MyFavoriteSubroutine=Alert "TxMessenger Rules!"`
	`ButtonOnscreen1Script=RUN MyFavoriteSubroutine`
	In an attempt to prevent infinite loops, an arbitrary limit is set on the depth of `RUN` commands that will be executed.

`SAVE`	Saves the current message or form (with current field contents) to the message list if not already saved. The contents of password fields are not saved.

`SAVE AS`	***Not available. This is reserved for possible future use.*** Asks the user for a filename using the standard save dialog. If the user doesn't cancel, the current form or message is saved as a text file using the path and filename specified.
`SAVE AS "`*text*`"`	***Not available. This is reserved for possible future use.*** Where "*text*" is a filename which can include a path The current form or message is saved as a text file using the path and filename specified.
`SELECT MESSAGE ALL`	Selects all the messages in the message list.
`SELECT MESSAGE DRAFT`	Selects all the draft messages in the message list.
`SELECT MESSAGE NAK`	Selects all the messages in the message list that were sent but negatively acknowledged by the host.
`SELECT MESSAGE NONE`	Deselects all the messages in the message list.
`SELECT MESSAGE READ`	Selects all the read messages in the message list.
`SELECT MESSAGE RECEIVED`	Selects all the received messages in the message list.
`SELECT MESSAGE SENDING`	Selects all the sending messages (but not negatively acknowledged) in the message list.
`SELECT MESSAGE SENT`	Selects all the sent messages in the message list.
`SELECT MESSAGE` *number*	Selects the message specified in the message list.

Note: All of the `SELECT TEXT` commands work as described regardless of INI settings. That is, if the INI says that only fields can be selected, these commands ignore the INI settings and select as commanded. (The `EditSelectionStyle` setting only restricts selections made by the user dragging the mouse.)

Peculiar selections can be formed programmatically. There is no guarantee that the selections will be handled well if the user then shift-selects. If desired, call `SELECT TEXT NONE` at the end of a script to get rid of odd selections.

SELECT TEXT commands not available in versions prior to v1.6.

`SELECT TEXT` *horizontal vertical*	Selects the character at the specified position. (Previously selected text remains selected.)

`SELECT TEXT` *horizontal vertical* `TO` *horizontal vertical*	Selects all characters from the first specified position to the second specified position. (Previously selected text remains selected.)
`SELECT TEXT ALL`	Selects all text (including read-only and fields).
`SELECT TEXT ALL FIELDS`	Selects all fields. (Previously selected text remains selected.)
`SELECT TEXT FIELD`	Selects the entire field that the caret is currently on. (Previously selected text remains selected.)
`SELECT TEXT FIELD "`*fieldName*`"`	Selects the entire field named (can also be a number). (Previously selected text remains selected.)
`SELECT TEXT NONE`	Deselects any selected text.
`SET` *settingName* `TO CARET FIELD NAME`	Sets the contents of the specified setting to the name of the field (if any) that the caret is at.
`SET` *settingName* `TO CARET FIELD NUMBER`	Sets the contents of the specified setting to the number of the field (if any) that the caret is at.
`SET` *settingName* `TO CARET HORIZONTAL`	Sets the contents of the specified setting to the horizontal location of the caret. This can be useful to move the caret back after positioning it elsewhere.
`SET` *settingName* `TO CARET VERTICAL`	Sets the contents of the specified setting to the vertical location of the caret. This can be useful to move the caret back after positioning it elsewhere.
`SET` *settingName* `TO CLIPBOARD`	*[Not available in versions prior to v1.6]* Sets the contents of the specified setting to the text contents of the clipboard.

`SET` *settingName* `TO DESK`	Sets the contents of the specified setting to the text contents of the form or message currently on the desk (if any). The content of password fields isn't stripped out.
`SET` *settingName* `TO DESK AS FORMNAME`	Sets the contents of the specified setting to the text contents of the message currently on the desk if it refers to an existing two letter internal formname. Otherwise, the setting is set to empty.
`SET` *settingName* `TO MESSAGE` *number*	*[Not available in versions prior to v3.5]* Sets the contents of the specified setting to the text contents of the message specified. Particularly useful in the MessagesReceivedScript in conjunction with `IF CONTAINS`, `TRUNCATE`, or `TRIM` commands.
`SET` *settingName* `TO "`*text*`"`	Sets the contents of the specified setting.
settingName `= "`*text*`"`	Shorthand version of above. Important note: This is not identical to the way you would change a setting by typing into the settings file! In this case, you must use quotes around everything after the equals sign! If you don't use quotes, the text will be interpreted as a *settingName* or *number*.
`SET` *settingName* `TO LENGTH "`*text*`"`	*[Not available in versions prior to v2.1]* Sets the contents of the specified setting to the number of characters in text.
`SET` *settingName* `TO FIELD "`*fieldName*`"`	Sets the contents of the specified setting to the contents of the specified form field. Numbers can be used instead of a field's name. The contents of password fields aren't stripped out.
`SET` *settingName* `TO SELECTION`	*[Not available in versions prior to v1.6]* Sets the contents of the specified setting to the selected text. The contents of the clipboard will not be altered.
`SET` *settingName* `TO SHOWN`	*[Not available in versions prior to v2.0]* Sets the contents of the specified setting to the name of the box that is currently displayed.
`SET` *settingName* `TO SCREEN MESSAGE`	*[Not available in versions prior to v2.0]* Sets the contents of the specified setting to the text displayed by the 'Scrolling Marquee' screen saver. Feature not guaranteed.

SET *settingName* TO VOLUME	*[Not available in versions prior to v2.0]* Sets the contents of the specified setting to the current volume level.
SET FIELD "*fieldName*" TO "*text*"	Sets the contents of the specified form field. When a text message or clear screen is present on the desk (instead of a form), SET FIELD 1 TO "text" sets the entire contents of the desk.
SET FIELD "*fieldName*" TO FIELD "*fieldName*"	Sets the contents of the specified form field to the contents of another specified form field. The content of password fields isn't stripped out.
SET SCREEN MESSAGE TO "*text*"	*[Not available in versions prior to v2.0]* Sets the text for the 'Scrolling Marquee' screen saver for the next time the screen saver is started. Feature not guaranteed.
SHUTDOWN	*[Not available in versions prior to v2.5]* Shuts down Windows and powers off the computer. Since this is operating system and hardware dependent, this command may not work completely, if at all.
SHOW "*boxName*"	Displays specified box or window. See "*boxName*" in the parameters section of this chapter for a complete list of different screens that can be displayed.
SHOW FILE	Displays a standard Windows OS open file dialog for the user to select a text, form, or message file. The file is then displayed.
SHOW FILE "*text*"	Displays the text, form, or message file specified with a pathname. Without a pathname, the file must be in the forms folder.

SHOW FORM "*formName*"	Displays the form with the specified form name. A setting can override this command and transmit the desk if there's a message or form with fields on the desk. If overriding isn't desired, use SAVE;CLEAR;SHOW FORM "formName". Leave off the SAVE if you don't want the current contents to be stored (like a two-letter form name typed on the desk). Here's how the Forms button is programmed by default:
	`SET InternallyReservedAndDeleted TO DESK AS FORMNAME;`
	`IF InternallyReservedAndDeleted EMPTY;`
	`SHOW "FORMS";`
	`ELSE;`
	`CLEAR;`
	`SHOW FORM InternallyReservedAndDeleted;`
	`ENDIF;`
	`DELETE InternallyReservedAndDeleted;`

SHOW MESSAGE	Displays the currently selected message (if any).
SHOW MESSAGE *number*	Displays the message with the specified ID.
SHOW NEXT	Displays the next message or form depending on which was last displayed.
SHOW NEXT IN "*boxName*"	If the box name is "Messages", then the next message (if any) is displayed. If the box name is "Forms", then the form after the currently selected form is displayed.
SHOW NEXT DRAFT	Displays the next oldest draft message (or wraps to newest if none found).
SHOW NEXT MESSAGE	*[Not available in versions prior to v1.3]* Displays the next message (upwards) in sort order.

SHOW NEXT UNREAD	Displays the next unread message (if any).
SHOW NEXT UNREAD IN "*boxName*"	If the box name is "Messages", then the next unread message (if any) is displayed. If the box name is "Forms", then the form whose internal (two-character TX) name that follows the currently displayed form's internal name is displayed. This "Forms" box algorithm emulates that on the legacy TX application.
SHOW PREVIOUS	Same as SHOW NEXT except this shows the previous.
SHOW PREVIOUS DRAFT	Displays the next newest draft message (or wraps to oldest if none found).
SHOW PREVIOUS IN "*boxName*"	Same as SHOW NEXT IN except this shows the previous.
SHOW PREVIOUS MESSAGE	*[Not available in versions prior to v1.3]* Displays the previous message (downwards) in sort order.
SLEEP *number*	Pauses (gives up CPU time) application for number of milliseconds specified.
SORT MESSAGES "*text*"	"" clears the sort order to internal default, eliminating prior sort criteria history. "*" makes the sort order customer standard (as defined in the settings file). "R" (received time), "Q" (queue), "P" (priority), "S" (subject), or "M" (message) becomes the primary sort criteria and lowers the prior sort criteria to lesser significance. If the prior, primary sort criteria were the same as the one specified, it toggles the primary sort order between ascending and descending. Only a single character can be specified as a parameter. To specify a complete and exact sort order string, use the SET command on the sort order setting.
SPEAK	***Not available. This is reserved for possible future use.*** Speaks the current message or form if speaking is available.

SPEAK "*text*"	***Not available. This is reserved for possible future use.*** Speaks the specified text if speaking is available.
START "*text*"	Where "*text*" is a filename which can include a path. The file will be started as though the user double-clicked on it in the file manager. This is useful for starting other applications, opening folders, or opening documents.
START "*text*" "*text*"	*[Not available in versions prior to v1.6]* Where the first "*text*" is the filename and/or path for an application and the second "*text*" contains parameters for the application. This is useful for starting an application with parameters.
SUBTRACT *settingName* BY *number*	Converts setting to a number, subtracts number from it, and stores the result back into setting.
TEST PARSER "*text*"	The "text" is evaluated as a Boolean. If true, the output that follows is the interpretation of the script without actual execution. If false, execution of the script resumes. This command is for testing the application's script interpreter, and may or may not be enabled in release versions of the application.
TRANSMIT	Transmits the current message or form as though the user had pressed the transmit button.
TRANSMIT "*text*"	Transmits the text as though the user had chosen New, typed the text, and then pressed the transmit button.
TRANSMIT CODED *number*	Transmits a code number (0-255 decimal) which is shorthand for a previously agreed upon message.
TRANSMIT EMERGENCY	Transmits an emergency signal.
TRANSMIT FORM REQUEST "*text*"	Requests, from the host, the form with the two-letter internal name specified.
TRANSMIT GPS	*[Not available in versions prior to v2.0]* Transmits the contents of the GPSData setting as non-compound GPS data. Does nothing if the GPSData setting is empty or deleted. The GPSData setting will be cleared after transmission. *Requires a valid GPS license number to be entered.*

`TRANSMIT GPS` *"text"*	*[Not available in versions prior to v2.0]* Transmits the text (or contents of a setting) as non-compound GPS data. Does nothing if the text or setting is empty. Doesn't clear the setting after transmission, even if the setting is GPSData. *Requires a valid GPS license number to be entered.*
`TRANSMIT MESSAGE` *number*	Transmits the message ID specified.
`TRANSMIT RAW` *"text"*	*[Not available in versions prior to v2.5]* Transmits the variable length message specified by *"text"*, bypassing the message list, GPS appending, and the usual TX processing / settings. Since no TX processing is performed, a full TX header must be specified. To allow binary values, the text is processed using the same technique as when reading text-file-based TX forms from the disk. That is, first all carriage returns and linefeeds are removed. Then, all hexadecimal values within <> are converted to their binary values.
`TRANSMIT RAW FILE` *"text"*	*[Not available in versions prior to v2.5]* Same as TRANSMIT RAW except that the message to send is read from the file named *"text"*.
`TRANSMIT STATUS` *number*	Transmits a status number (0-255 decimal) which represents the user's current state.
`TRIM` *settingName*	*[Not available in versions prior to v1.2]* Removes white space from the beginning and ending (and condenses multiple white space) of the current value of the setting and stores the trimmed value back into the setting.

`TRUNCATE` *settingName* `TO` *number*	*[Not available in versions prior to v2.1]* Removes all characters after the number specified and stores the value back into the setting. Example:
	`test = "Samuel";`
	`TRUNCATE test TO 3;`
	`ALERT test;`
	Produces "Sam".

`TRUNCATE` *settingName* `TO` *number number*	*[Not available in versions prior to v2.1]* Removes all characters before the first number and after the last number specified and stores the value back into the setting. Example:
	`Chuck = "The duck quacked";`
	`TRUNCATE Chuck TO 5 8;`
	`ALERT Chuck;`
	Produces "`duck`". Out-of-range values are fine to use (no error). Example:
	`TRUNCATE Chuck TO 5 100000;`
	Produces "duck quacked".

TYPE "*text*"

Enters the text similarly to a user typing it. Unlike other commands that execute immediately, each character of the text is queued and won't appear until after the script has finished and the application has time to process them. Therefore, don't expect to be able to sequence this command between other commands. A good rule of thumb is that a script should contain only the type command by itself or the type command should be the last command of the script.

A type command is effective for filling in a string where the user's caret is currently located (PASTE is better). To place field information reliably without the limitations of the type command, use the SET FIELD command. When a text message or clear screen is present on the desk (instead of a form), SET FIELD 1 TO "text" sets the contents of the desk.

Because characters in the type command are fed into the keyboard queue, any modifier keys (control, shift, or alt) in effect at the time of reading the key may be applied to the characters.


Because of the limitations and side effects of the TYPE command, use PASTE "text" or SET FIELD commands instead.

Beginning with v2.5, the following characters generate special keys:

\xD8	backspace
\xD9	tab
\xDD	enter
\xEB	esc
\xF1	page up
\xF2	page down
\xF3	end
\xF4	home
\xF5	left arrow
\xF6	up arrow
\xF7	right arrow
\xF8	down arrow
\xFD	insert key
\xFE	delete

`UPPERCASE` *settingName*	*[Not available in versions prior to v2.5]* Converts all lowercase letters from ‘a’ to ‘z’ of the current value of the setting to uppercase letters ‘A’ to ‘Z’ and stores the result back into the setting.
`VOLUME` *number*	*[Not available in versions prior to v2.0]* Sets the volume level to the specified number. Valid values are 0 through 10.
`WIGGLE`	*[Not available in versions prior to v2.0]* Moves the cursor, which deactivates some screen savers. For those screen savers that are responsive to this command, the terminal will remain locked if the screen saver is password protected. If the `WIGGLE` command doesn’t work for some screen savers, try `TYPE "\t"`, which may act as though the user pressed the tab key.
`WRITE FILE` *"text" "text"*	*[Not available in versions prior to v2.5]* Appends the second “text” specified to the end of the file specified by the first “text” (with or without a path). A file marked in Windows as read-only won’t be affected. If the file specified doesn’t exist, it will be created. This function can be useful for creating an independent log file or copying certain data to a text file for use by another program.  Use this command with care. If you add text to the wrong file, the file can be corrupted.
`WRITE SETTINGS`	*[Not available in versions prior to v1.2]* Writes the settings file to disk with the current values. This command is not normally necessary, as the settings file is always written at exit.

8 Text Form Creation Tutorial

This section is a tutorial that explains how to use the form and message formatting sequences in order to make text forms. Example files referred to in this tutorial are included with TxMessenger shipping media and can be viewed using a text editor, such as the Windows Notepad accessory.

8.1 Introduction

TxMessenger uses text-file-format forms to make the creation and modification of forms an easier task. Thus, a form can be created simply by typing the text and saving it as a text file.

Most forms consist of both fields (places where data can be input) and read-only text (not user modifiable). TxMessenger uses formatting sequences that determine the way the text appears and where the fields are located. The formatting sequences that TxMessenger supports are demonstrated in this tutorial and listed individually later in this document.

In most sections of the tutorial, a form is created and then added to throughout. When the tutorial describes "adding lines to the form," that refers to adding lines to the form created in that section. Each section of the tutorial uses a separate form (indicated under the section's title), and some use several forms. All forms created in the tutorial have been included with TxMessenger for reference.

All parameters, that is, places where the form designer would place a number or letter, are indicated by *italic* text, and the type of value that should be put in that place is described in the tutorial. Most of the formatting sequences described in this tutorial start with "<1B>." "1B" is the hexadecimal value for the escape sequence, and, when set off by < >, the sequence indicates to TxMessenger that a formatting sequence is following. Other hexadecimal values such as <0F> are used for sequences. Knowledge of hexadecimal values is not required to make forms. All sequences that use hexadecimal values are explained, and it is only the syntax of the sequence that is important to know when writing forms. In most cases, if "0" appears, the character is a zero, not an upper case letter "O," unless otherwise specified.

8.2 Basic Sequences

(Form made in this section: `basics.txt`)

To start off, you will make a very simple form and open it in TxMessenger. Open Notepad or any other plain text file editor and type the lines:

```
TXBA<00><10><00>
This is
my first form.
```

Now, save the file as basics.txt in TxMessenger's Forms folder and open TxMessenger. Click the Forms button and then click the icon for the form you just made to see how it looks in TxMessenger. Notice that TxMessenger got the text right, but it didn't place the text on two separate lines. This introduces the first formatting sequence, Carriage Return.

8.2.1 Starting a New Line - Carriage Returns

The Carriage Return sequence enables you to end a line of text and start a new one. This sequence is "<0D>" (note that the "0" is a zero, not the letter "O"). In the form, the second line should end after the word "is", therefore, just following that, place the formatting sequence, such that the form now looks like this:

```
TXBA<00><10><00>
This is<0D>
my first form.
```

Save the form again and open it in TxMessenger. Now, it displays the form with two lines of text.

8.2.2 Blank Lines – Multiple Carriage Returns

What do you do if you want a blank line of text in-between the lines of text? Try putting two Carriage Returns together so that the second and third lines of the form now read:

```
This is<0D><0D>
my first form.
```

Save the form and open it again. Notice that there is no blank line—TxMessenger purposely ignored the second Carriage Return. This is due to the way in which the TX-protocol requires handling of consecutive Carriage Return sequences. Blank lines are intentionally eliminated in order to fit as much text onto the screen at once.

When TxMessenger encounters a Carriage Return, it sets the next place to put text as the first position of the next line; however, if a Carriage Return is encountered at the first position of a line (or at the very beginning of a form), TxMessenger does not move the position down another line. In order to make multiple Carriage Returns, a different character such as a space must be placed before the second Carriage Return.

Put a space between the carriage returns in the example form:

```
This is<0D> <0D>
```

TxMessenger displays a blank line now. But, not in this next case:

```
This is<0D>
<0D>
my first text form.
```

It may appear as if the Carriage Returns are separated, but TxMessenger ignores the separate lines that you have made in Notepad. In effect, there is no character separating the two Carriage Returns and the second one will be ignored. Simply stated, wherever a line should end, a Carriage Return must be placed.

Although not required, it is easier to maintain a form if blank lines are placed on individual lines with space characters at the start of the line, such as:

```
This is<0D>
 <0D>
my first text form.
```

8.2.3 Faster Editing

Forms can be edited while TxMessenger is running.

1. Run TxMessenger.
2. Open the desired form in the Windows Notepad accessory.
3. Make a change to the form.
4. Choose Save from the File menu in Notepad.
5. Leave the form open in Notepad.
6. Switch to TxMessenger.
7. Click the Forms button in TxMessenger.
8. Click the form in TxMessenger to display it.

You can switch back and forth between TxMessenger and Notepad while both are running and continue making changes and redisplaying (steps 7 and 8) the form. This makes form editing convenient and provides almost immediate results.

8.2.4 TX-Protocol Form Header

Before proceeding further, making a header in a form must be discussed. In the TX-protocol, every form must be designated with a two-character internal name specified at the start of the form. The syntax for the most common header is TX*cc*<00><10><00>, where *cc* is the two-character (usually letters) internal form name being specified. The syntax of this sequence need not be understood, but this sequence is necessary for TX-protocol messaging applications to handle a form properly. For TxMessenger to properly recognize the example form, insert a header as the first line:

```
TXBA<00><10><00>
This is<0D>
 <0D>
my first text form.
```

Note that this sequence specifies the two-character internal name BA for the form, which was chosen because this chapter covers some basics about forms, so the BA stands for basic. Any two-character internal name could have been chosen, however. In subsequent chapters, when a new form is created, a suggested form name will be given. This name must be put in the header line, and the header line must be put at the start of a form.

8.3 Basic Field Sequences

(Form made in this section: `fields.txt`)

So far, the forms described have been composed entirely of read-only text, that is, there was not any place for the user to type into the form.

8.3.1 Starting and Ending Fields

To allow input in a form, TxMessenger uses Fields—places where text can be typed. The Start Field and End Field sequences begin and end a Field, respectively. All of the spaces after a Start Field sequence will be spaces in a field, until an End Field sequence is encountered. The Start Field sequence is <1B>[0o, and the End Field sequence is <1B>[1o. To see how this works, make a new form with the lines below, give it a header with the name FI (as shown), save it, and view it in TxMessenger:

```
TXFI<00><10><00>
field <1B>[0o     <1B>[1o
```

Notice that, now, after the word "field," there is a Field where text—numbers and/or punctuation—can be typed. This is not the only kind of Field available, there are several other types of fields, one that only accepts text and punctuation, one that only accepts numbers and punctuation, and one that only can be filled in by the host. A field that only accepts numbers is started with the Start Numeric Field sequence, <1B>[3o. A field that only accepts alphabetic characters is started with the Start Alphabetic Field sequence, <1B>[4o. A field that can only be filled in by the Host is started with the Start Guarded Field sequence, <1B>[8o. Note that all fields are ended with the same End Field sequence, that is, there is one generic End Field sequence for all types of Fields. Also note that the fields also accept punctuation, as noted earlier, and the punctuation that is accepted is ( ) . + - * / . To see these types of Fields, add lines to the form you created to make it look like this (note the <0D> sequences at the end of every line):

```
TXFI<00><10><00>
field <1B>[0o     <1B>[1o<0D>
numbers only <1B>[3o     <1B>[1o<0D>
alphabetics only <1B>[4o     <1B>[1o<0D>
guarded field <1B>[8o     <1B>[1o<0D>
```

Try typing in each field, and notice that there appears to be nothing in the form where the Guarded Field should appear, because only a host computer can fill in text in that Field.

8.3.2 Field Default Text

There is something interesting about the fields that has not been addressed yet. What if you put some text in-between the Start Field and End Field sequences, instead of using just spaces? Doing so puts this text into the field as "default data." This is information that will be in the field already, but which can be typed over when filling out the form. This is useful for something like the year, as it can be filled in already. Modify the form so that it now looks like this:

```
TXFI<00><10><00>
unprotected field <1B>[0odata  <1B>[1o<0D>
numerics only <1B>[3o1111<1B>[1o<0D>
alphabetics only <1B>[4odata<1B>[1o<0D>
guarded field <1B>[8odata<1B>[1o<0D>
```

The fields now have default data that can save time in filling in fields that, for the most part, will be filled in with the same information.

Unlike text typed by the user, the default data is not restricted or filtered based on the field type. That is, numeric fields can have alphabetic characters as default data. However, once deleted or altered by the user, the user cannot retype the default characters back into the field.

With the sequences covered in this section, you will be able to create forms with simple fields. Fields are used in many forms to obtain input from the user. The next section covers more advanced sequences to use in fields.

8.3.3 Carriage Returns in Fields

 Never use Carriage Returns in fields. The TX-Protocol is ambiguous as to the results to expect and all the messaging applications (Motorola TX for Windows, Motorola WaveSoft-Link, and TxMessenger) handle Carriage Returns differently.

```
TXCR<00><10><00>
Avoid this: <1B>[0oHow long is<0D>this
field?<1B>[1o
```

8.4 Advanced Field Sequences

(Form made in this section: `advanced fields.txt`)

This section covers sequences that make fields more adaptable for specific types of input.

8.4.1 Password Fields

Two sequences that are very useful for forms such as login screens are the Password sequences, Start Password and End Password. Password sequences hide text typed into some or all characters of a field. The text typed is not shown but is represented by a character such as an asterisk, "*", so that the password cannot be read by others.

The Start Password and End Password sequences are respectively <1B>12h and <1B>[12l (lower-case letter "L"). To see what a password field is like, make a new form with the line shown below, give the form a header with the title AF (as shown) and open it in TxMessenger:

```
TXAF<00><10><00>
password <1B>[12h<1B>[0o     <1B>[1o<1B>[12l<0D>
```

Notice that the Start Password sequence did not begin the field; the Start Field and End Fields sequences were still required. The Start Password and End Password sequences modify how fields appear, they don't start or end fields.

8.4.2 Repeat Character Compression

A sequence that is often encountered with long fields is the Repeat Character sequence. This sequence can be used anywhere in a form, not just in fields. It is most often used with fields that are long. The Repeat Character sequence allows you to repeat a character a specified number of times. The format is *c*<1B>[*n*b, where the italic letters denote values that you fill in—*c* denotes the character to be repeated, and *n* denotes the number of times to repeat the character. Note that the first character of the sequence already counts as 1 character to display, meaning that, to display a character 10 times using the Repeat Character sequence, the *n* part of the sequence should only be set to 9. To see how this works, add the lines below to the form you've been working on:

```
repeat character <1B>[0o <1B>[9b<1B>[1o<0D>
          TEN character long field<0D>
```

Notice what is happening in the form: a field is started (<1B>[0o), then a space appears, the sequence to repeat the space nine times follows, then the field is ended. This makes for a total of ten spaces in the field, the first space in the file and nine repeated spaces. Finally, add the lines below to the form to see the Repeat Character sequence in other cases:

```
Repeat Character<1B>[4b<0D>
Repeat Character z<1B>[4b<0D>
```

These demonstrate the repeating of characters in a form, which can be used for various purposes, including use with graphics characters (discussed later) to create borders or other graphics.

A downside of the Repeat Character sequence is that it is harder to read and modify by someone editing the form file. Unless the form file is being transmitted from the host to the mobile, the number of bytes saved in the form definition is usually not worth the effort of using a Repeat Character sequence. The user won't see a difference on the screen and TxMessenger automatically compresses text transmitted to the host.

A limitation of the Repeat Character sequence is that it takes 5 bytes at minimum. There are no savings between XXXXX and X<1B>[4b. Because of complexity and overhead, don't use the Repeat Character sequence except for 6 characters or more. In addition, the number of times a character can be repeated is limited to a maximum of 255. Multiple repeat sequences must be included if more than 255 characters are needed

8.4.3 Setting Cursor Position

The Set Cursor Position sequence sets the position of the cursor in or to a field. If the specified position is not in a field, the cursor will not be placed in read-only text. If the Set Cursor Position sequence is specified more than once in a form, only the last sequence is honored.

The Set Cursor Position Sequence is <1B>[*v*;*h*H. In this sequence, *v* denotes the vertical position of the cursor and *h* denotes the horizontal position of the cursor. To see how this works, add the line below to the form you've been working on:

```
<1B>[2;16H
```

This puts the cursor in the first half of the second line (assuming your example form has a field there) when the form is displayed. This can be used in such cases as when the first few fields contain default data and normally would be left unchanged, meaning that the usual field to be modified is not the first field. By placing the cursor in a specific field of a submitted form, the host can indicate which field contains a mistake or missing information.

8.4.4 Uppercase and Lowercase

Two more advanced sequences that modify fields are the Auto-Text sequences. Upper Case Auto-Text and Lower Case Auto-Text change the text entered to upper or lower case, respectively. If text is typed in a part of a field where an Auto-Text sequence is present, it will be automatically converted to the case specified. However, if, for instance, the left half of a field is Upper Case Auto-Text, and lower case text is typed there and then moved to the right half of the field (ex. by inserting numbers before it), the lower case text, when it exits the Auto-Text portion of the field, will return to lower case. The Upper Case Auto-Text sequence is <1B>[Au, and the Lower Case Auto-Text sequence is <1B>[Al (lower-case letter "L"). To end Auto-Text, the End Auto-Text sequence can be used, <1B>[An. To see how Auto-Text affects fields, add this line to the form you've been working on:

```
auto-text <1B>[0o<1B>[Au     <1B>[Al
<1B>[An     <1B>[1o<0D>
```

Experiment with typing in this field now and watch what happens to the text. Note that the first five characters should always be upper case, the next five characters should always be lower case and the last five characters should be whichever case is typed.

8.4.5 Checkbox Field

An additional Auto-Text sequence that is useful in certain cases is the Checkbox sequence. When used like the other Auto-Text sequences, the Checkbox sequence makes a space in a field that is marked with an "X" when any alphanumeric is entered in it. To see this, add this line to the form you've been working on:

```
checkbox <1B>[0o<1B>[Ax <1B>[An<1B>[1o<0D>
```

With this, an "X" is always placed in the field, no matter what is entered.

Note: In the above example, the checkbox is an unprotected field, so an "X" is always placed in the field. If the checkbox was a numeric only field, then a "1" would be placed in the field. If the checkbox was a lowercase only field, then a lowercase "x" would be placed in the field.

8.4.6 Field Names

The Field Name sequence allows a name to be associated with a field. This allows the scripting language to refer to a field by name, regardless of the field's position or order in the form. If all the fields in a form are named, the fields themselves can be moved around, lengthened, shortened, removed, or otherwise manipulated and scripts or external programs (like script enabled magnetic-stripe readers) can still place the field data in the correct fields!

The Field Name sequence can be placed anywhere in-between the Start Field and End Field sequences that define a field, EXCEPT at the first position after the Start Field sequence. The syntax of the sequence is <1B>[SF"*ccc*...", where *ccc*… is the field name to be specified. To try the Field Name sequence, add the line below to the form:

```
ST: <1B>[0o        <1B>[SF"State"<1B>[1o
```

Now, using the scripting language, the name of the field that is defined in the line above will be accessible.

In the Script Tester window, try this script:

```
SET FIELD "State" TO "Illinois"
```

Notice that the read-only text that appears before the field, "ST:", is never used by TxMessenger as the field name. Read-only text is only for the user's benefit and can contain any "name" or "label" that is meaningful to

the user. Only fields containing the Field Name sequence are actually named as far as TxMessenger is concerned.

With the various sequences presented in this section, you will now be able to customize fields in your forms and create forms that are more advanced.

8.5 Graphic Sequences

(Forms made/referenced in this section: `graphics.txt`, `all graphics.fm`)

TxMessenger has several built-in character sets that enable line graphics to be displayed in place of characters. Line graphics can be used for making boxes around text, emphasizing text, or separating certain text.

8.5.1 Enable and Disable Graphic Characters

To turn on graphics characters in a form, the Enable Graphics Characters sequence is used. All characters following this sequence are displayed as graphics, until a Disable Graphics Characters sequence is encountered. The Enable Graphics Characters sequence is <0F> and the Disable Graphics Characters sequence is <0E>. To see graphics characters, make a form with the following in it, giving it a header with the title GR (as shown):

```
TXGR<00><10><00>
<0F>ABCDEFGHIJKLMNOPQRSTUVWXYZ<0D>
abcdefghijklmnopqrstuvwxyz<0D>
```

When viewing the form, some characters will be replaced with graphics. As mentioned earlier, several different character sets can be used. Note that Graphics Characters need not be turned off if the rest of the form following an Enable Graphics Characters sequence is supposed to be composed entirely of graphics characters.

Graphics Characters are useful for making boxes that set off text or fields and are useful for making line drawings such as maps.

8.5.2 Graphic Character Set

Various devices and applications have displayed different images for various graphic characters. Because the same character represents a different on-screen shape depending on the device or application, TxMessenger supports 5 different graphic character sets to match the set used by the original device or application.

The TxMessenger.ini setting `GraphicsCharacterSet` specifies the default graphic character set to use unless explicitly set otherwise in a form or message.

```
GraphicsCharacterSet=None          // Set 0
GraphicsCharacterSet=Secondary     // Set 1
GraphicsCharacterSet=Host          // Set 2
GraphicsCharacterSet=Terminal      // Set 3
GraphicsCharacterSet=All           // Set 4
```

To otherwise explicitly set the character set used in a form, the Graphics Set sequence, <1B>[G*n* can be used. In this sequence, *n* denotes the character set number. The numbers 0 through 4, inclusive, represent valid character set numbers. Graphics set 4 contains many graphics characters, therefore, set the characters in the form to use this set by adding the line below:

```
<1B>[G4
```

Graphics characters can be turned on and off anywhere in a form, enabling these line graphics to be put in a form if desired, but only one graphics set can be specified for a form. If more than one Graphics Set sequence is encountered, TxMessenger will only honor the last one in the form.

Experiment with the other Graphics sets (numbers 0 through 3) to see the graphics that are contained within (use “All Graphics.fm”). See the form “Character Set” for a complete list of characters that have a graphic counterpart in set 4.

8.5.3 Pictures

The Insert Picture sequence allows you to insert a picture into a form or message and has the syntax <1B>[SP"*cccccccc.ccc*", where *cccccccc.ccc* denotes the file name of the picture (for example “Car.bmp”).

The character position where the sequence appears determines the location of the top-left corner of the picture. Use spaces (or other text) before the Insert Picture sequence to move the picture to the right. Since the width and height of the text characters determine the starting point of the picture, resizing the window or changing the number of columns in the form will move the picture to a different starting point.

Unlike text, pictures aren’t resized for different window widths.

The picture is drawn over text or fields; therefore, if using images in a form, enough space must be left in the text to accommodate the picture.

Add the following line to the form to see the insertion of a picture:

```
<1B>[SP"FormMotorola.bmp"<0D> <0D>
```

Note that if a picture does not exist, TxMessenger will not display an error message but will leave a blank space where the top left corner of the picture would appear in the form.

8.6 Customizing Appearances

(Forms made in this section: `appearances.txt`, `bg colors.txt`, `fg colors.txt`, `borders.txt`, `appearances2.txt`)

This section describes various methods that enhance the way in which a form appears when displayed.

8.6.1 Highlight, Underline, and Blink

Some simple sequences for enhancing text are the Highlight, Underline and Blink sequences. The Start Highlight and End Highlight sequences are <1B>[5m and <1B>[0m, respectively. The Start Underline and End Underline sequences are <1B>[U1 and <1B>[U0 (in "U0" there is a zero, not a letter "O"), respectively. The Start Blink and End Blink sequences are <1B>[7m and <1B>[!, respectively. To see the effects of these sequences, make a new form, give it a header with the name A1 (as shown) and add the lines shown below:

```
TXA1<00><10><00>
<1B>[5mHighlight this<1B>[0m<0D>
<1B>[U1Underline this<1B>[U0<0D>
<1B>[7mBlink<1B>[!<0D>
```

This shows how TxMessenger highlights, underlines and blinks text. Highlight, underline and blink are useful for emphasizing important parts of a form.

8.6.2 Foreground Text Color

A good way of differentiating text is to make it different colors. The Foreground Color and Background Color sequences will change the colors displayed for text (foreground) and the background. When one of these sequences is encountered, all text following the sequence will have its color changed, until another one of these sequences is encountered.

The sequence to set the foreground color can be typed with the syntax <1B>[F*cc* or <1B>[F*nnnnnn*. In the first case, *cc* denotes a predefined color, designated by two lower case letters. In the second case, *nnnnnn* denotes the hexadecimal value of a color, enabling the use of any color in a form. Hexadecimal colors are represented by six alphanumeric characters.

To see how foreground colors work, make a new form, give it a header with the name FG (as shown) and add the following lines:

```
TXFG<00><10><00>
<1B>[FreRED<0D>
<1B>[ForORANGE<0D>
<1B>[FyeYELLOW<0D>
<1B>[FgnGREEN<0D>
<1B>[FblBLUE<0D>
<1B>[FmgMAGENTA<0D>
<1B>[FbrBROWN<0D>
<1B>[FbkBLACK<0D>
<1B>[FwhWHITE<0D>
<1B>[FgrGRAY<0D>
<1B>[FlgLIGHT GRAY<0D>
<1B>[FigMIDDLE GRAY<0D>
<1B>[FnaNAVY<0D>
<1B>[FmoMAROON<0D>
<1B>[FcyCYAN<0D>
<1B>[Fnm <0D>
```

When you open this form, you will see that the color of the text on each line matches the color stated. Note that there is a “normal” or “return to default” sequence at the end.

8.6.3 Background Text Color

Now, make another form, like this one, but give it a header with the name BG (as shown) and, instead of changing the foreground color, change the background color using these lines:

```
TXBG<08><10><00>
<1B>[BreRED<0D>
<1B>[BorORANGE<0D>
<1B>[ByeYELLOW<0D>
<1B>[BgnGREEN<0D>
<1B>[BblBLUE<0D>
<1B>[BmgMAGENTA<0D>
<1B>[BbrBROWN<0D>
<1B>[BbkBLACK<0D>
<1B>[BwhWHITE<0D>
<1B>[BgrGRAY<0D>
<1B>[BlgLIGHT GRAY<0D>
<1B>[BigMIDDLE GRAY<0D>
<1B>[BnaNAVY<0D>
<1B>[BmoMAROON<0D>
<1B>[BcyCYAN<0D>
<1B>[Bna <0D>
```

Notice how, just like with the Foreground Color sequence, the Background Color sequence changes the background of all spaces, until another Background Color sequence is found.

An edge appears around the form that is navy, and the space below the form is also navy. The last background color applied to text in a form is what the edge color is set to. In this case, the background is set to navy, and then there is a space, which the navy color is applied to. This changed the edge color to navy. The default background can be set with:

```
<1B>[Bnm <0D>
```

8.6.4 Border Text Color

The markers for each character of a field, called borders, and any underlining can also be customized. The borders default to a bucket-like state with a green color. The Border Color sequence works exactly like the other Color sequences, except it changes the color of the borders and underlining. The Border Color sequence is <1B>[E*cc* or <1B>[E*nnnnnn*. Again, *cc* denotes a predefined color and *nnnnnn* denotes a hexadecimal value for a color. To see the effects of these sequences, make a form with the following lines:

```
TXA2<00><10><00>
normal <1B>[0o     <1B>[1o<0D>
red <1B>[Ere<1B>[0o     <1B>[1o<0D>
still red <1B>[0o     <1B>[1o<0D>
yellow <1B>[Eye<1B>[0o     <1B>[1o<0D>
green <1B>[Egn<1B>[0o     <1B>[1o<0D>
blue <1B>[Ebl<1B>[0o     <1B>[1o<0D>
```

8.6.5 Field Character Border Styles

The border style can also be changed, as the borders don't have to look like buckets. The Border Style sequence, which changes the style of the border, has the syntax <1B>[D*n*, where *n* is a number between 0 and 5 inclusive. To see what the different border styles look like, add border style sequences to the form, so that it looks like this:

```
<1B>[D0normal <1B>[0o     <1B>[1o<0D>
<1B>[D1red <1B>[Ere<1B>[0o     <1B>[1o<0D>
<1B>[D2still red <1B>[0o     <1B>[1o<0D>
<1B>[D3yellow <1B>[Eye<1B>[0o     <1B>[1o<0D>
<1B>[D4green <1B>[Egn<1B>[0o     <1B>[1o<0D>
<1B>[D5blue <1B>[Ebl<1B>[0o     <1B>[1o<0D>
```

Notice that Border Style number 0 is a blank border and does not show anything, but the cursor will still operate in the same fashion in the field. The Border Color and Border style allow the borders in a form to be customized such that, for instance, the borders for required fields in a form could be colored red. Also, for a field that needed a few lines of text, the borders could be changed so that they are less obtrusive on the text being typed and show that the field is a free-form text area.

8.6.6 Comments in Forms

The Insert Comment sequence allows a person making a form to insert a comment in the form that will not be displayed to the user, but which a person editing the actual text file will see.

The Insert Comment sequence has the syntax <1B>[{*cccc...*}, where *cccc*… denotes the comment. When editing the form, these comments can be seen, but when the form is displayed in TxMessenger, there will be no evidence that comments exist in the form. Make a new form containing the following lines and observe that TxMessenger gives no indication of the existence of the comment:

```
TXA2<08><10><00>
<1B>[{this is a comment}
```

Comments are particularly nice for noting the time, date, type of change, and editor's name when modifying a form. This can be useful to keep a running log of changes within the form itself.

Because they take up room in the form without user visible effect, comments should not be used in the final forms transmitted over-the-air from the host to the mobile device. The comments will just waste airtime.

8.6.7 Form Width – Number of Columns

In a form, TxMessenger defaults to using 40 columns (or the value of the setting TextNumberOfCharactersPerLine) to display the text. This can be overridden for specific forms or messages by using the Number of Columns sequence. The syntax for this sequence is <1B>[N*nn*W, where *nn* represents a number from 1 to 160. This is useful because the size of the text is automatically adjusted based on the number of columns, meaning that bigger or smaller text can be used by using fewer or more columns, respectively. To see this, add the following line to the form:

```
<1B>[N20W
Bigger text can be seen here
```

The Number of Columns sequence must appear before any text or else the line breaks may not be correct. Only one Number of Columns sequence may be used per a form or message.

8.7 Forms Screen

(Form made in this section: `forms screen.txt`)

You have probably noticed that with the forms created thus far in this tutorial, the Forms screen in TxMessenger seems a bit plain, and the titles chosen by TxMessenger for forms aren't always clear. On the Forms screen, TxMessenger tries to choose an icon that is fitting for a form, but often, like with the tutorial forms, no appropriate icon can be found and a default icon is used. Also, TxMessenger tries to guess at a good name for each form, but the tutorial forms do not have any clearly defined titles, so the names TxMessenger chooses are a bit unclear. To improve the look of the Forms screen, form titles and picture icons should be explicitly specified within forms.

8.7.1 Form Title

The Change Title sequence lets you designate a title for a form. This sequence can be put at any point in a form, and if multiple instances of this sequence appear, only the last one will be recognized. The Change Title sequence is written <1B>[ST"*cccc...*", where *cccc...* denotes the new title name.

Make a new form, type in the lines shown below, save it and open it in TxMessenger.

```
TXFS<08><10><00>
Unclear forms<0D>
with editing<0D>
can become clear<0D>
```

Note that TxMessenger chooses the title "Unclear forms" for this form. Now, add the line below to the file:

```
<1B>[ST"Title Experiment Form"
```

Now, on the Forms screen, TxMessenger displays the title "Title Experiment Form." In this way, if you wish to have a title that is more appropriate than the title TxMessenger chooses for a form, you can change the form's title to a suitable name.

8.7.1.1 Form Title with Key

If a key has been configured to display a form, the form title can be modified to indicate a key in the Forms screen.

```
<1B>[ST"Vehicle Inquiry\n-F1-"
```

Adding key information to all of the form titles in the Form screen really makes form access convenient and easy to remember.

8.7.2 Form Picture Icon

The Change Icon sequence allows the changing of the forms icon (the picture that is displayed in the Forms screen). The syntax is <1B>[SI"*cccccccc.ccc*", where *cccccccc.ccc* is a filename.

To see how this sequence works, add the line below to the form:

```
<1B>[SI"FormMotorola.bmp"
```

Now, on the Forms screen, the Motorola logo will be displayed as the icon. In this manner, the icon for any form can be changed. If the icon file is not found, TxMessenger will not display an error message but will display its default icon for the form.

8.7.3 Form Tool Tip Help

When the cursor is placed over a form's icon in the Forms screen, tool tip help (if enabled) can be shown for a form. The syntax of Tool Tip Help sequence is <1B>[SH"*ccc*…", where *ccc*… represents the text of the tool tip.

To see a tool tip, add the following line to the form:

```
<1B>[SH"This is a tool tip."
```

Now, open TxMessenger and hold the cursor over the form (the text name of the form or the icon for the form) and the tool tip will appear (assuming that tool tips are enabled in TxMessenger). Tool tips enable short messages about a form to be accessible in this way.

8.7.4 Arrive and Read Sounds

In addition, a sound can be set to play when TxMessenger receives or displays a form (or data for a form or a message). The Arrival Sound and Read Sound sequences accomplish these, respectively. The syntax of the Arrival Sound sequence is <1B>[SA"*cccccccc.ccc*" and the syntax for the Read Sound sequence is <1B>[SS"*cccccccc.ccc*". To see the effects of these sequences, add the following lines to the form:

```
<1B>[SA"keyclick.wav"
<1B>[SS"startup.wav"
```

Now run TxMessenger. When TxMessenger loads this form, you will hear the keyclick.wav sound. When you open this form from the Form screen (or any button or script programmed to open this form), you will hear the startup.wav sound. In this way, sounds can be associated with forms.

8.7.5 Variations of a Form that the Host Processes as the Same Form

A form's internal name is transmitted to the host along with the corresponding fields when the user presses the transmit button. This name determines what the host looks up or does. The internal form name of a form must be unique; TxMessenger will honor only one form with that internal name.

In order to have multiple variations of a form, all which transmit the same name to the host for processing, each form must be given a unique form name in the TX-Protocol header, but the same transmit form name.

The Internal Form Name sequence specifies the internal form name that is used when the form or fields are transmitted. This allows for multiple forms with the same name to be sent in (but the forms themselves could have different font sizes, field defaults, pictures, or colors). The syntax of this sequence is <1B>[TF*cc*, where *cc* is the form name to be sent to the host when the form is transmitted. The line below can be added to the form:

```
<1B>TFAA
```

With this, the form name AA will be sent to the host instead of the normal form name (if one is set). This allows TxMessenger to have several different forms that it submits as the same form, meaning that similar forms could be made that get treated by the host as the same form.

8.8 Unused Sequences

There are several sets of unused sequences which, when encountered in a form, will be ignored by TxMessenger. These sequences have been specified in this manner so that future versions of TxMessenger can remain backward compatible with this version yet at the same time add more functionality in forms.

TxMessenger will not display “garbage” in a form when it encounters such a sequence; rather, it will simply ignore it.

Unused Strings have the syntax <1B>[S*c*"*cccc*…" and enable future string sequences to be placed in a form.

Unused Auto-Text sequences have the syntax <1B>[A*c* and can be used in future versions to have other Auto-Text in a field.

Unused Extended Sequences have the syntax <1B>[X*cc* and are available for any sequences that might be added in future versions.

Unused Number Sequences have the syntax <1B>[N*nnC*. Unused Scripts have the syntax <1B>(*cccc*…). As with a comment, if TxMessenger encounters any of these sequences, it will show no sign that the sequence exists.

 **Important:** Do not use these sequences! The Motorola TxMessenger development team reserves them for future use.

9 Form and Message Formatting Sequences

This section specifies the formatting sequences available for text messages and forms. Before reading this section, reading the Text Form Creation Tutorial (section 8) is recommended, as it explains use of the sequences and includes example forms to aid in understanding syntax rules.

9.1 Table of Formatting Sequences

Numbers between <> represent hexadecimal values. Type the <> and the numbers exactly as shown within forms that are in text file format (.txt). For binary (.fm, .fmt, and so on) file formats or in use over the air, the <> and number should be replaced by a single byte character of the value specified by the number.

"X"s in the Allowable Usage column represent formatting sequences that both Motorola TX and Motorola TxMessenger support. "M"s represent sequences that only Motorola TxMessenger supports.

Name	Actual Sequence	Allowable Usage		
		Messages	Forms	Form Data
Basic Formatting (TX):				
Set Cursor Position	<1B>[*v*;*h*H	X	X	X
Carriage Return	<0D>	X	X	
Form Feed (not supported)	<0C>			
Repeat Character (compression)	*c*<1B>[*n*b	X	X	X
Record Separator	<1E>			X
Start Field (formerly Start Unprotect)	<1B>[0o	M	X	
End Field (formerly Start Protect)	<1B>[1o	M	X	
Start Numeric Field	<1B>[3o		X	
Start Alphabetic Field	<1B>[4o		X	
Start Obsolete Field	<1B>[o			
Start Guarded Field	<1B>[8o		X	
Enable Graphics Characters (formerly called Shift In)	<0F>	M	X	
Disable Graphics Characters (formerly called Shift Out)	<0E>	M	X	
Start Highlight	<1B>[5m	M	X	
End Highlight	<1B>[0m	M	X	
End Highlight (obsolete)	<1B>[m			
Start Password (formerly called Disable Echo)	<1B>[12h		X	
End Password (formerly called Enable Echo)	<1B>[12l		X	
Start Blink	<1B>[7m	M	X	
Start Blink (obsolete)	<1B>[7!			
End Blink	<1B>[!	M	X	

Control Sequence Name	Sequence	Allowable Usage		
		Messages	Forms	Form Data
Advanced Formatting (TxMessenger):				
Foreground Color	<1B>[F*cc* or <1B>[F*nnnnnn*	M	M	
Background Color	<1B>[B*cc* or <1B>[B*nnnnnn*	M	M	
Border Color	<1B>[E*cc* or <1B>[E*nnnnnn*	M	M	
Border Style	<1B>[D*n*		M	
Start Underline	<1B>[U1	M	M	
End Underline	<1B>[U0	M	M	
Insert Comment	<1B>[{*cccc...*}	M	M	
Insert Picture	<1B>[SP"*cccccccc.ccc*"	M	M	
Upper Case Auto-Text	<1B>[Au		M	
Lower Case Auto-Text	<1B>[Al		M	
Checkbox	<1B>[Ax		M	
End Auto-Text	<1B>[An		M	
Graphics Set	<1B>[G*n*	M	M	
Change Title	<1B>[ST"*cccc...*"	M	M	
Change Icon	<1B>[SI"*cccccccc.ccc″*	M	M	
Number of Columns	<1B>[N*nn*W	M	M	
Arrival Sound	<1B>[SA"*cccccccc.ccc*"	M	M	
Read Sound	<1B>[SS"*cccccccc.ccc*"	M	M	
Field Name	<1B>[SF"*ccc*…"		M	
Tool Tip Help	<1B>[SH"*ccc*…"	M	M	
Transmit Form Name	<1B>[TF*cc*		M	
Unused Strings	<1B>[S*c*"*cccc*…"	M	M	
Unused Auto-Text	<1B>[A*c*	M	M	
Unused Extended	<1B>[X*cc*	M	M	
Unused Number	<1B>[N*nnC*	M	M	
Script	<1B>(*cccc*…)	M	M	

9.1.1 Formatting Sequences in Drafts and Inbound Transmissions

When a text message or fields of a form are saved or transmitted to the host, the formatting sequences are stripped out. That is, only plain text is saved or transmitted. For example: If a message with color and underlines is received and read by the mobile user and then resaved to the message list, the color and underlines are removed.

9.2 Set Cursor Position

Format: <1B>[*v*;*h*H

Parameters: *v*: Vertical line number—1 or 2 ASCII digits.
h: Horizontal column position—1 or 2 ASCII digits from 1 to 40.

Description: This sequence sets the position of the cursor in or to a field.

Notes:

This sequence is optional.

Cursor position has no effect on the placement of text in the form.

When multiple cursor position sequences are found, only the last one is honored.

If the cursor position specified is not within a field, the cursor is placed in the first position of the first field.

9.3 Carriage Return

Format: <0D>

Parameters: None.

Description: The Carriage Return is treated as a Carriage Return/Line Feed combination. The cursor is moved to the first column of the next line on the screen. If the cursor is already at the first position of the display line, it does not move.

Notes:

9.3.1 Blank Lines

Multiple contiguous Carriage Returns are treated as a single Carriage Return. In order to move down multiple lines (or insert blank lines), it is necessary to place a character (such as a space) between each Carriage Return entry.

9.3.2 Carriage Returns Within Field Definitions

Carriage Returns should not appear in unprotected (field) portions of forms. The TX-Protocol is ambiguous as to whether the field should be extended to the end of the line, or stop and continue at the beginning of the next line. Motorola TX, Motorola WaveSoft-Link, and TxMessenger handle the situation differently. TxMessenger can be configured (in the setting `FieldDefinitionCarriageReturnProcessing`) to try to match the total field length of TX or WaveSoft-Link.

Instead of using a Carriage Return in field definitions, use spaces to define the field length explicitly.

9.4 Form Feed

Format: <0C>

Parameters: None.

Description: TxMessenger ignores Form Feed characters.

Notes:

In other programs, a form feed starts a new page of text. The operator could use the Tab key, arrow keys or scroll bar to navigate between pages in a message. The programmer may use Form Feeds anywhere in the message. If the Form Feed is in line 3, a three-line message is displayed. Placement of the next Form Feed in line 20 causes lines 4 to 20 of the message to be displayed. TxMessenger uses a scrollable window to display all of the text of a message or form in the specified scrollable area, as opposed to using separate pages to display the text of a message or form.

9.5 Repeat Character (compression)

Format: c<1B>[nb

Parameters: c: Character to repeat.
n: Number of times to repeat character (ASCII digit or digits).

Description: The Repeat Character sequence repeats the previous character a specified number of times. The number of times specified to repeat the character does not include the original copy of the character (Ex. “AAAA” in a form can be represented by the sequence “A<1B>[3b”). The number of times a character can be repeated is limited to a maximum of 255. Multiple repeat sequences must be included if more than 255 characters are needed.

9.6 Record Separator

Format: <1E>

Parameters: None.

Description: The Record Separator separates strings of unprotected text characters on inbound and outbound transmissions. In outbound messages, the Record Separator delimits the data that will be placed into each unprotected field of a format. Two Record Separators in a row cause no data to be entered into a field. In inbound messages, the Record Separator delimits the unprotected data from each field in a form or formatted text message. If the unprotected data sent to the terminal is shorter than the field in which the data is to be placed, the downloaded data is left-justified in the unprotected field. If the transmitted text is longer than the unprotected field, the text is truncated. Protected text is not normally transmitted. One Record Separator is sent for each protected field (except the first). Attributes within protected fields (blinking protected, graphics protected) do not change the number of Record Separator characters sent. The first Record Separator in a message appears after the first unprotected field.

9.7 Starting and Ending Fields (formerly called Start Unprotect and Start Protect)

Format:

Start Field:	<1B>[0o
End Field:	<1B>[1o
Start Numeric Field:	<1B>[3o
Start Alphabetic Field:	<1B>[4o
Start Guarded Field:	<1B>[8o
Start Obsolete Field:	<1B>[o

Parameters: None.

Description: Formatting sequences specify how message contents will be organized into data fields, and what attributes particular data fields may possess. Editable fields in a form begin with a Start Field sequence and end with an End Field sequence. An Alphanumeric Start Field sequence enables the user to input alphabet characters, numbers and punctuation ( ) . + - * / in a field. The Start Numeric Field sequence restricts the accepted input in a field to numbers and punctuation, and the Start Alphabetic Field sequence restricts the accepted input to alphabet characters (upper or lower case) and punctuation. A Start Guarded Field may be filled in by the host, but not edited by the terminal user. Default information can be placed in a field between the Start Field and End Field sequences, if desired.

Notes: The Start Obsolete Field sequence <1B>[o is supported to ensure backward-compatibility but this sequence should not be used.

9.8 Enable / Disable Graphics Characters (formerly called Shift In / Out)

Format: Enable Graphics Characters: <0F>
Disable Graphics Characters: <0E>

Parameters: None.

Description: Various character sets are included with TxMessenger to enable the display of line graphics in forms. Any characters between Enable Graphics Characters and Disable Graphics Characters sequences will be replaced with the graphic assigned to the character in the current character set. The current character set can be specified in a form using the Graphics Set sequence.

Notes: These formatting sequences are not supported in Motorola TX for Windows or in Motorola WaveSoft-Link.

9.9 Start / End Highlight

Format:

Start Highlight:	<1B>[5m
End Highlight:	<1B>[0m
End Highlight (obsolete):	<1B>[m

Parameters: None.

Description: All fields following a Start Highlight sequence will be set to highlight, until an End Highlight sequence is found.

Notes: The default condition of the display is non-highlighted.
The End Highlight (obsolete) formatting sequence <1B>[m is supported for backward-compatibility but should not be used.

9.10 Start / End Password (formerly called Disable / Enable Echo)

Format: Start Password: <1B>[12h
End Password: <1B>[12l (lower-case letter “L”)

Parameters: None.

Description: When in a field, if a key is hit, the key will normally be displayed in the field. For a password, where it is preferable that the keys hit are not displayed, the Start Password and End Password sequences can be used. After a Start Password sequence, the operator can enter characters into an unprotected text field, but the characters are not displayed. The display cursor progresses to the next character position as each key is pressed. All unprotected fields following the Start Password formatting sequence are treated as password fields, until an End Password sequence is reached.

9.10.1 Passwords in Drafts and Sent Queue

For security reasons, the contents of password fields are stripped out before being saved to the message list. Therefore, when the user types a password in a form, saves the form, and recalls the form, the password field is empty.

9.11 Start / End Blink

Format:

Start Blink:	<1B>[7m
Start Blink (obsolete):	<1B>[7!
End Blink:	<1B>[!

Parameters: None.

Description: All fields following a Start Blink sequence will be set to blink, until an End Blink sequence is found.

Notes: The Start Blink (obsolete) sequence is supported for backward-compatibility, but it should not be used. The Start Blink and End Blink sequences are not supported in TX for Windows or WaveSoft Link.

9.12 Foreground / Background / Border Color

Format:

Foreground Color:	<1B>[F*cc* or <1B>[F*nnnnnn*
Background Color:	<1B>[B*cc* or <1B>[B*nnnnnn*
Border Color:	<1B>[E*cc* or <1B>[E*nnnnnn*

Parameters: *cc*: Specify a defined color using two lower-case alphabet characters.
nnnnnn: Specify any color using hexadecimal values.

Description: The color settings can be used to enhance forms and emphasize information in forms. The Foreground Color specifies the color of all characters following the formatting sequence that are displayed in a form, including the color of the text in a field. The Background Color specifies the background color of the display for all positions following the sequence. The Border Color specifies the color of the Borders of field spaces (the lines that define positions which are unprotected) and the color of underlining. All characters and spaces in the form after the color sequence will be changed, until another color sequence of the same type is encountered. TxMessenger has a set of defined colors that can be specified with two letter abbreviations, as shown in the table on this page. For any other color, the six digit hexadecimal value for the color must be used.

Notes: TxMessenger displays a one-character edge around a form. The color of this edge is defined by the last background color appearing in the form. To change this color, the last sequence in the form should change the background to the color you desire, and a space and then an Carriage Return (<0D>) should follow. This makes the edge color appear differently without affecting the colors displayed in the form.

For a defined color, the two-character color parameter must be specified in lower-case (Ex. To make text blue, the sequence "<1B>[Fbl" would be used).

Color	Abbreviation	Hexadecimal Value
Black	bk	0x000000
Blue	bl	0x0000FF
Brown	br	0x663300
Cyan	cy	0x00FFFF
Gray	gr	0xC0C0C0
Green	gn	0x00FF00
Light Gray*	lg	0xE0E0E0
Magenta	mg	0xFF00FF
Maroon	mo	0x990000
Middle Gray*	ig	0x808080
Navy	na	0x000099
Orange	or	0xFF6600
Red	re	0xFF0000
White	wh	0xFFFFFF
Yellow	ye	0xFFFF00

**not available in versions prior to v1.2*

Note: When using the six digit hexadecimal value anywhere other than in forms, reverse the red and blue values to get the desired color. For example, if you wanted the background of the Forms screen to be purple using the hexadecimal value `0x9933CC`, you would set `BoxFormsBackgroundColor=0xCC3399.`

9.13 Border Style

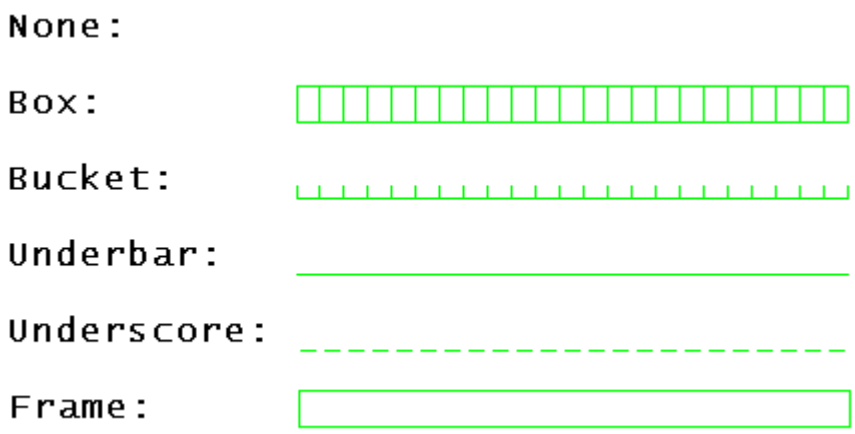
Format: <1B>[Dn

Parameters: n: Specifies a number from 0-5 where each number represents a different border style.

Description: TxMessenger defaults to a bucket-like border (the border is the unprotected text field indicator), but several other options are available. All borders in a form following this formatting sequence will be changed to the specified style, until another Border Style sequence is encountered. The border styles are defined as follows:

0 - None
1 - Box
2 - Bucket
3 - Underbar
4 - Underscore
5 - Frame

Here is how each border style appears in TxMessenger:



9.14 Start / End Underline

Format: Start Underline: <1B>[U1
End Underline: <1B>[U0

Parameters: None.

Description: The Underline formatting sequence allows forms to have underlined text. Text following a Start Underline sequence will be underlined until an End Underline sequence is encountered.

9.15 Insert Comment

Format: <1B>[{cccc...}

Parameters: cccc...: A comment.

Description: The Insert Comment formatting sequence enables comments to be inserted in forms. When TxMessenger displays a form, comments are not shown and there is no indication of whether or not comments exist. Programmers can use comments when designing forms.

9.16 Insert Picture

Format: <1B>[SP"cccccccc.ccc"

Parameters: cccccccc.ccc: Specifies the picture file name, in 8.3 format (For example: “Motorola.bmp” or “Car.bmp”).

Description: In a form, TxMessenger allows pictures to be displayed. When this sequence is inserted in a form, the picture referred to will be displayed with its upper-left corner being the space where the formatting sequence was placed. If the image file is not found, no image will be displayed. There is no error message displayed when an image is not found. TxMessenger simply leaves a blank space in the form where the upper-left corner of the image would have appeared.

Note: Pictures to be used must be put in the Pictures folder, within TxMessenger’s folder.

TxMessenger does not wrap text around a picture, rather, it draws all pictures over text; therefore, adequate space must be left in the text for the picture.

9.17 Upper Case / Lower Case Auto-Text, Checkbox (in field)

Format:

Upper Case Auto-Text:	<1B>[Au
Lower Case Auto-Text:	<1B>[Al (lower-case letter "L")
Checkbox:	<1B>[Ax
End Auto-Text:	<1B>[An

Parameters: None.

Description: The Upper Case Auto-Text and Lower Case Auto-Text sequences can be applied to an entire field (section of unprotected text) or to parts of a field such that only the case specified is allowed. Characters entered are automatically shown in the proper case. If, for example, a field is half Upper Case Auto-Text and half normal text, then, if lower case letters are entered in the upper case portion, upper case letters will be displayed. However, if characters are inserted at the beginning of the field, pushing the original lower case letters out of the Upper Case Auto-Text portion, the characters will revert to the form in which they were entered (be it lower case, upper case or mixed). These sequences are useful for fields that prompt for such information as a two-letter state abbreviation, where the characters entered could be automatically capitalized.

The Checkbox sequence is a specific type of Auto-Text. Where the Checkbox sequence is used, no matter what alphanumeric is hit, an "X" will always be sent back to the host. Like the other Auto-Text sequences, this sequence can be applied to an entire field or to parts of a field (though it is designed to be applied to a field that is one character long). This is useful when checkbox-type input is necessary.

Notes: When using these sequences, an End Auto-Text sequence must be placed where the Auto-Text should end, including at the end of a field. If an End Auto-Text sequence is not used in a field, then all subsequent fields will have the Auto-Text (or Checkbox) attribute, until an End Auto-Text sequence is encountered.

9.18 Graphics Set

Format: <1B>[Gn

Parameters: n: Specifies a number from 0-4 where each number represents a different Graphic Set.

Description: Different sets of characters can be used in TxMessenger, and the Graphics Set formatting sequence allows you to specify which set will be used in a form. The different sets of characters have line graphics that can be used in forms. Any characters specified as graphics (characters appearing after a <0F> sequence but before a <0E> sequence) will be shown with the corresponding graphics character in the Graphics Set specified.

If the sequence appears more than once, only the last is honored.

9.19 Change Title

Format: <1B>[ST"cccc…"

Parameters: cccc…: Text of replacement title

Description: When forms are listed, TxMessenger takes a guess as to what a form name ought to be. At times, this form name may be unclear, may not fit in the title area, or may just seem inappropriate. To fix this, the Change Title sequence is available so that a programmer can specify a title for a form, overriding TxMessenger's guess at a suitable form name.

If the sequence appears more than once, only the last is honored.

Note: The quotes shown in the syntax of this sequence are not to denote text, but rather the form name must appear in quotes. For example, to specify that a form should be called "Example Form," the formatting sequence <1B>[ST"Example Form" would be used.

9.20 Change Icon

Format: <1B>[SI"cccccccc.ccc"

Parameters: cccccccc.ccc: Specifies the icon file name, in 8.3 format (Ex. “FormCar.bmp”).

Description: When TxMessenger displays forms, it takes a guess as to what a good icon for each form would be. In many cases, it chooses an appropriate icon, but in some cases, it cannot find an appropriate icon and therefore displays a default icon. The Change Icon sequence is designed for this case and for any other case in which a programmer wishes to change the icon displayed. Any Windows bitmap that is placed in the Pictures folder, within the TxMessenger folder, can be used as an icon for a form or forms. If this sequence is used, TxMessenger searches the Pictures folder for a file with the name specified and displays it as the icon. If no such file is found, TxMessenger uses its best guess at a good picture for an icon, but there is no “file not found” error message displayed. The Change Icon sequence allows programmers to ignore the default icon pictures altogether and use another set of icons, if desired.

If the sequence appears more than once, only the last is honored.

9.21 Number of Columns

Format: <1B>[NnnW

Parameters: nn: Specifies a number from 1 to 160 for the number of columns.

Description: In a message or form, TxMessenger defaults to using 40 columns to display the text, but a different number of columns can be used. TxMessenger automatically adjusts the text size to fit the column, meaning that having fewer columns across the screen will give bigger text in the message or form.

If the sequence appears more than once, only the last is honored, but earlier sequences may affect padding and alignment at the time they were in effect.

9.22 Arrival / Read Sound

Format: Arrival Sound:<1B>[SA"cccccccc.ccc"
Read Sound: <1B>[SS"cccccccc.ccc"

Parameters: cccccccc.ccc: Specifies the sound filename (for example: “beep.wav”).

Description: Sounds can be associated with forms using the Arrival Sound and Read Sound sequences. When the Arrival Sound sequence is used, the specified sound is played when TxMessenger receives the message or form. When the Read Sound sequence is used, the sound specified is played when the message or form is opened for reading in TxMessenger.

If the sequence appears more than once, only the last is honored.

Notes: Sound files used must be wave format and have the “.wav” extension. Sounds should be placed in the Sounds folder in TxMessenger’s folder. If TxMessenger does not find the specified sound, no sound will be played.

9.23 Field Name

Format: <1B>[SF"ccc…"

Parameters: ccc…: The field name to be specified.

Description: This specifies a name for a field in a form. The scripting language built into TxMessenger uses field names in order to reference certain fields. The Field Name sequence allows the administrator to specify a name for the field instead of having to refer to the field by a number. This number is based on the field's position relative to other fields in the form. For example, the first field encountered is field 1; the second field is field 2, and so on.

Naming fields makes scripts more readable and portable. Because unique and recognizable names can be given to fields, the fields can be moved on the form or added to the form. Although the field's number would change, its name remains constant.

The formatting sequence must be placed inside of the field for which the name is to be specified; otherwise, the sequence is ignored.

Notes: The sequence CANNOT be placed at the first position in the field (right after the Start Field sequence) because it will then be ignored. There must be a space or default data before the sequence, but the sequence can fall anywhere after the first position or after any default data in a field (including directly before the End Field sequence).

9.24 Tool Tip Help

Format: <1B>[SH"ccc…"

Parameters: ccc…: The text of the tool tip help.

Description: In TxMessenger, on the Messages and Forms screens (the screen listing all of the messages and the screen listing all of the forms, respectively), Tool Tip Help can be associated with a message or form using the Tool Tip Help sequence. This is a way to give information about a message or form that is particularly useful for users who are beginners with TxMessenger.

If the sequence appears more than once, only the last is honored.

9.25 Internal Form Name

Format: <1B>[TFcc

Parameters: cc: The form name to be transmitted (two characters).

Description: If several forms are used that should be processed by the host in the same way, then the forms can all use the same form name when being sent to the host. The Internal Form Name sequence changes the name of the form that is transmitted to the host when a form is transmitted. This sequence is not necessary unless the form name being specified is different than the form name specified in the header. This is useful if forms with variations are to be processed by the host as the same form, such as a basic identification form and a thorough identification form.

If the sequence appears more than once, only the last is honored.

9.26 Unused Strings / Auto-Text / Extended Sequences / Number Sequences / Scripts

Format:

Unused String:	<1B>[S*c*"*cccc*…"
Unused Auto-Text:	<1B>[A*c*
Unused Extended Sequences:	<1B>[X*cc*
Unused Number Sequences:	<1B>[N*nn*C
Unused Scripts:	<1B>(*cccc*…)

Parameters: *cccc*…: Specifies a string.
c: Specifies an alphabetic character.
cc: Specifies two alphabetic characters.
nn: Specifies a number from 01 to 99.

Description: The unused sequences are sequence sequences that TxMessenger will not display when encountered. These unused sequences have been built into TxMessenger so that in future versions of the program, new sequences may be made while still remaining compatible with this version of TxMessenger. In the future, if new sequences are defined using these Unused Sequences, TxMessenger simply will ignore them and will not display "garbage" on the screen when reading a form because it is unable to interpret the sequence.

 Important: Do not use these sequences! The Motorola TxMessenger development team reserves them for future use.

9.27 Script

Format: <1B>[(ccc…)

Parameters: ccc…: The embedded script.

Description: Beginning with TxMessenger v1.6, scripts can be embedded in messages (or forms). When the message is displayed on the desk, the script is executed. Reloading the message causes the script to run again.

Because the right parenthesis character, “)”, ends the script sequence, the right parenthesis character can’t appear in the embedded script. However, the character can be generated in quoted text by using \041.
For example: `<1B>[(ALERT "( Hi \041")`

If the script sequence appears more than once, only the last is honored.

10 Other Screen

The Other button in TxMessenger displays a screen of infrequently used functions. This provides the capability of adding many additional customized buttons without using up valuable screen space (like Onscreen buttons).

Instead of programming a peculiar function key (such as Ctrl-Shift-F9) for a rarely needed feature, it is more user-friendly to add an Other button with an easily identifiable picture, name, and Tool Tip description.

10.1 Settings for Other Buttons

Other buttons are created in the same way as are Status and Coded buttons.

10.1.1 Other Button Label

The label setting defines the button's title, which is the text displayed beneath the button's picture on the Other screen.

```
ButtonOther#Label=Example
```

10.1.2 Other Button Script

The script setting defines the action to be performed when the button is clicked (see the Script Command Reference in section 7 of this document).

This setting may be missing or blank, but then the button won't do anything when clicked.

```
ButtonOther#Script=ALERT "Example button
activated!"
```

10.1.3 Other Button Help

The help setting describes the text to display when the cursor is over the button and Tool Tips are turned on (`ToolTipsActive=TRUE`).

This setting may be missing or blank, but then no help text would appear for the particular button.

```
ButtonOther#Help=Click me to display an alert.
```

10.1.4 Other Button Picture

The picture setting defines the filename of the picture to display for the button in the Other screen. The picture may be any of those that are built-in or a picture that exists in the Pictures folder (see Pictures folder in section 5.4 of this document).

This setting may be missing or blank, in which case the OtherDefault.bmp picture is displayed.

```
ButtonOther#Picture=OtherEmergency.bmp
```

10.1.5 Other Button Sort Order

The sort order setting defines the order in which the button is to appear in relation to all the Other buttons. Buttons usually have a sort order value of 0 (zero), in which case they are all sorted alphabetically. Buttons with a lower sort order (negative numbers) appear before and buttons with a higher sort order (1 and up) appear after. Buttons are sorted alphabetically within their sort order group.

This setting is usually missing or blank, in which case the sort order is 0.

```
ButtonOther#SortOrder=0
```

10.1.6 Other Button Numbering

Each group of settings for a particular button uses a number from 1 to 300. This number must be unique and must be the same for each setting of that particular button. The number is not related to the sort order or to where the button appears in the Other screen.

```
ButtonOther93Label=Example
ButtonOther93Script=ALERT "Example button
activated!"
ButtonOther93Help=Click me to display an alert.
ButtonOther93Picture=OtherEmergency.bmp
ButtonOther93SortOrder=0
```

The number 93 was chosen arbitrarily for the above example, but it could be any number between 1 and 300. When adding to TxMessenger.ini, it is probably less confusing just to start from the number 1 and work up.

10.2 Changing Button SortOrder

A button can be placed before others by giving it a lower sort order setting. For the example button:

```
ButtonOther93SortOrder= -1000
```

A button can be placed after others by giving it a higher sort order setting. Such as:

```
ButtonOther93SortOrder=2
```

10.3 Launching an Application or Opening a File

One of the most convenient uses of a button on the Other screen is to launch an application or open a file. There's a really easy way to create a button to do so:

1. Run TxMessenger.
2. Click the Other button.
3. From Windows file manager, drag the desired file onto the Other screen (the central portion of the TxMessenger window that contains all the Other buttons).
4. TxMessenger automatically creates a button with the file's name and icon!

As an example, dragging Notepad.exe resulted in the following settings being added to TxMessenger.ini:

```
ButtonOther8Help=
ButtonOther8Label=NOTEPAD.EXE
ButtonOther8Picture=UseAssociatedIcon
ButtonOther8Script=START "C:\\\\NOTEPAD.EXE"
ButtonOther8SortOrder=8
```

The help setting should be manually modified (or added if it isn't there) to provide Tool Tip help.

The label setting can be made a little more descriptive by changing it to "`Notepad`".

The picture setting reveals a special "picture name" of UseAssociatedIcon, which indicates TxMessenger should try to find the icon indicated by the path in the Script.

The script setting has two backslashes for every one backslash normally present in a pathname. This is because a backslash in quotes normally indicates a special character. Two backslashes are required to indicate a single backslash in quotes.

The sort order setting places the button at the end of the current Other screen. This setting can be removed to alphabetically sort the button in with the rest.

It isn't necessary to drag-and-drop to create a button in the Other screen. These settings can be added by hand to TxMessenger.ini. Drag-and-drop is simply easier.

To delete a drag-and-drop-created button, exit TxMessenger and remove the newly added ButtonOther settings in TxMessenger.ini.

10.4 Settings for Built-in Other Buttons

All of the built-in buttons in the Other screen are created by default before TxMessenger reads TxMessenger.ini. Therefore, the attributes associated with these buttons can be overridden by a script or setting in TxMessenger.ini.

The built-in buttons are numbered as follows:

Title	Number	
About TxMessenger	1001	
Exit	1002	
Coded Message	1003	
Keys	1004	
Modem Status	1005	(in v1.2 or earlier)
Form Names Shown / Hidden	1006	
Tool Tips Show / Hidden	1007	
Sounds Off / On	1008	

10.4.1 Configuring Single Press Access to Other Functions

Rather than clicking the Other button and then clicking a button on the Other screen (two clicks), a function key or Onscreen button can be scripted to perform the same action with only a single click or keypress.

The easiest way to reproduce the action of a built-in Other button is to reproduce its script. To see the script:

1. Type Ctrl-t to view the Tools Screen. Click the Display Current Settings button. (In TxMessenger v1.2 or earlier, Ctrl-t displays the Script Tester. In the Script Tester window, type `SHOW "Settings"` and press return.)

2. Scroll until the settings beginning with ButtonOther are displayed.

3. Change the Script setting for a key or onscreen button to match the ButtonOther#Script.

For example, to display the Keys screen, an onscreen button could be scripted like ButtonOther1004Script, which is `SHOW "Keys"`. See Key and Onscreen Button Configuration in section 13 of this document for instructions on configuring keys and onscreen buttons.

10.5 Disabling or Removing the Modem Status Button

If `Configure=TRUE`, simply right-click on the Modem Status button in the Tools screen and choose Delete from the popup menu.

In TxMessenger version 1.2 and earlier:

The Modem Status button can be disabled from the Other screen in TxMessenger by adding the following blank setting to TxMessenger.ini:

```
ButtonOther1005Script=
```

This script is equally effective and a little more informative:

```
ButtonOther1005Script=ALERT "Modem Status screen has
been disabled. Contact system administrator if
assistance is required."
```

Setting all the Modem Status button settings to blank in the TxMessenger.ini leaves a blank button on the Other screen, because the existence of a setting (even if blank) causes the Other button to be created. (Starting in TxMessenger version 1.2, Other buttons can be removed by simply deleting or setting the Label portion to blank)

The Modem Status button can be removed completely by changing the StartupScript setting to:

```
DELETE ButtonOther1005Label;
DELETE ButtonOther1005Script;
DELETE ButtonOther1005Help;
DELETE ButtonOther1005Picture;
DELETE ButtonOther1005SortOrder;
```

The StartupScript setting should also include the screen or page to show at startup, otherwise the splash screen is displayed on the desk until the user clears it or brings up a form. The StartupScript usually ends with something like:

CLEAR;CLEAR MORE;SHOW FORMS;

The remaining Other buttons can be removed or disabled in a similar fashion to the Modem Status button, by replacing 1005 with the desired button number.

10.6 The Exit Button

The Exit button is much easier to press on a touch screen than is the tiny close box in the upper-right corner of the window.

The Exit button is automatically hidden when the setting `Exit=NEVER`. However, the hiding feature may not operate correctly if the Exit button is renamed or another Other button is named Exit.

11 Colors

Most of the text colors and background colors in TxMessenger can be configured in settings. Color settings can be set to either the name of a color or a numeric value.

11.1 Color Names

Many popular colors are defined by name. Using a color name makes it easier to maintain a setting, since it is instantly obvious what color is being used.

```
ButtonMenuTextColor=WHITE
```

Color Name	Equivalent Numeric Value
BLACK	0x00000000
BLUE	0x00FF0000
BROWN	0x00003366
CYAN	0x00FFFF00
GRAY	0x00C0C0C0
GREEN	0x0000FF00
LIGHT GRAY	0x00E0E0E0
MAGENTA	0x00FF00FF
MAROON	0x00000099
MIDDLE GRAY	0x00808080
NAVY	0x00990000
RED	0x000000FF
ORANGE	0x000066FF
WHITE	0x00FFFFFF
YELLOW	0x0000FFFF

11.1.1 Using Quotes for Color Names

In a script, make sure to put quote marks around the color name, so as not to confuse it with another setting name.

```
ButtonMenuTextColor="BLACK"; /* In a script */
```

But for a single setting value in the settings file, don't use quotes.

```
ButtonMenuTextColor=BLACK  /* In settings file */
```

Quotes are standard in scripting languages and no quotes are standard in settings files. Although this can be confusing when configuring both, it provides compatibility with both popular standards.

11.2 Color Numeric Values

Any color can be defined using a numeric value. Although numeric values are less obvious than using a color's name, TxMessenger only defines a few colors by name. Using a numeric value is the only way to define certain colors.

Color numeric values always* start with 0x00…

Followed by two hexadecimal digits for the amount of blue (00 to FF, with FF being the most)…

Followed by two hexadecimal digits for the amount of green…

Lastly, followed by two hexadecimal digits for the amount of red.

No digits can be left off the right end. 0x00FF isn't the same as 0x00FF0000.

0x00000000 is black because it doesn't have any blue, green, or red. In other words, no color.

0x00FFFFFF is white because it has the most amounts of blue, green, and red mixed together. In other words, all color.

0x00FF0000 is maximum blue.

0x007F0000 is about half-maximum blue.

0x0000FF00 is maximum green.

0x000000FF is maximum red.

0x0000FFFF is maximum green and maximum red mixed together. In other words, maximum yellow!

*actually, colors don't need to be specified in hexadecimal. They can be specified using regular numbers (base 10) or even base 2. However, hexadecimal color values are easier to create since hexadecimal breaks the blue, green, and red into constant digit places.

11.2.1 Comparing Color Names and Color Numbers

Settings can be set to a color or number, but not compared between the two in a script. That is, if a setting is a color name, a script can't successfully compare it to a setting number (or vice-versa).

```
Color1="BLACK";
Color2=0x00000000;

IF Color1="BLACK";
     RESPOND "Color1 is the color black, specified by name.\x0D\x0A";
ENDIF;

IF Color2=0x00000000;
     RESPOND "Color2 is also the color black, but specified
numerically.\x0D\x0A";
ENDIF;

IF Color1!=Color2;
     RESPOND "But Color1 does not equal Color2.";
ENDIF;
```

11.3 Color Settings

All settings that affect colors have the word "color" somewhere in their name. Searching the electronic version of the settings documentation for the word "color" is the best way to find all of the settings that affect colors.

11.4 Day and Night Mode

When a message or form is displayed on the Desk, day mode uses the exact colors specified in the message or form, or if unspecified, in the settings.

Night mode applies an algorithm that detects a light background text color with dark foreground text color and swaps, darkens, or brightens the colors. No changes occur if the form or message already has a dark background, because night mode is intended to reduce the brightness of the light emitted from the display.

Picture colors remain unchanged in both day mode and night mode. This means that if a large picture contains a lot of white or other bright colors, it will increase the amount of light emitted by the device at night.

11.5 Replacing Default Day/Night Mode Button

The ColorMode setting determines whether the color settings are to be used exactly (day mode) or as to whether light backgrounds should be darkened (night mode).

In day mode, all text colors display exactly as specified. In night mode, all text colors on dark or medium text-backgrounds display exactly as specified. However, light text-backgrounds (such as white) darken and their foreground text lighten, similar to reversing or flipping the foreground / background color choices. This automatic method usually results in an overall darker screen.

Default text colors for night mode and even multiple variations of night and day mode can be specified. `ButtonCorner1Script` can be set to the following script (scrunch it onto a single line in TxMessenger.ini) to override the default day/night mode button.

```
IF ColorModeOverride=1;    /* Custom night mode */
       ADD ColorModeOverride BY 1;
       ColorMode="Night";
       ButtonMenuTextColorNightMode="BLACK";
       TextColorForeground="WHITE";
       TextColorBackground="BLACK";
ELSE IF ColorModeOverride=2;/* Extra night mode (if desired) */
       ADD ColorModeOverride BY 1;
       ColorMode="Night";
       ButtonMenuTextColorNightMode="GREEN";
       TextColorForeground="RED";
       TextColorBackground="BLACK";
ELSE;                      /* Custom day mode */
       ColorModeOverride=1;
       ColorMode="Normal";
       TextColorForeground="BLACK";
       TextColorBackground="WHITE";
ENDIF;
RELOAD DESK;
```

The TextColorForeground and TextColorBackground settings have no effect on the color of text explicitly colored with a formatting sequence within a form or message.

Remove `ButtonCorner1Script` from TxMessenger.ini to restore the default button.

There isn’t any way to specify a white (or light) background color that isn't affected by night mode. To prevent automatic flipping of text colors, ColorMode must be prevented from being set to "Night". For example, the corner button could be overridden (above) but always set ColorMode="Normal".

11.6 Background Colors and Pictures

All built-in form, status, coded, and other icon bitmaps are 75 pixels by 75 pixels, square. The edges and background are black to achieve the illusion of a smaller size or different shape (such as a circle) when displayed on the forms, status, coded, and other screens.

When that same picture is displayed in a form or message with a non-black background color (or reversed background color such as in day vs. night mode), the true rectangular shapes of the bitmap is seen.

TxMessenger doesn't alter colors in a picture, even if the colors don't match or blend into the background.

In summary, when designing or adding a custom picture that is to be used in the forms screen, status screen, coded screen, or other screen, use black as the background or edge color. When designing a picture for a form or message, use a frame or neutral edge color in the picture that looks attractive against the background color of the Desk. Test the picture in day and night mode.

12 Upgrading from WaveSoft-Link

 **Important:** Although TxMessenger is a direct upgrade for WaveSoft-Link, the products aren't identical.

TxMessenger supports most of the features of Motorola WaveSoft-Link 100 and 200. However, because TxMessenger provides a superior graphical user interface, powerful scripting language, and wider customization, there are unavoidable differences from WaveSoft-Link.

12.1 Scheduling Enough Time and Resources

Like any other software replacement, whether it is a web browser, word processor, or operating system, a certain amount of downtime, retraining, and incompatibility should be anticipated. Technicians and administrators should schedule time and resources to configure and certify TxMessenger and to reinstall updated versions if necessary.

Some customers may be content with the minimal configuration and feature set that matches WaveSoft-Link. However, TxMessenger has a number of wonderful improvements that are worth reserving enough time to allow the technician or administrator to configure them.

Simply mimicking WaveSoft-Link won't take full advantage of the user-friendly form, status, and other pictures. Nor will it provide more powerful multi-command function keys or special buttons in the Other screen. The forms themselves can also be improved with titles, color, graphics, more whitespace, and/or a bigger font. These touch ups can make the difference between a satisfied user and a happy, excited user.

12.2 Install Onto a Computer That Already Has WaveSoft-Link

To ease conversion, install TxMessenger onto a computer that already has a working version of WaveSoft-Link. TxMessenger doesn't install files or modify system resources that prevent or alter WaveSoft-Link's operation, and vice-versa.

When installed on the same computer, side-by-side comparisons can be made regarding forms, function keys, screens, and modem interaction. This also allows TxMessenger to find the existing configuration of WaveSoft-Link and invoke any automatic conversion capabilities present in that version of TxMessenger.

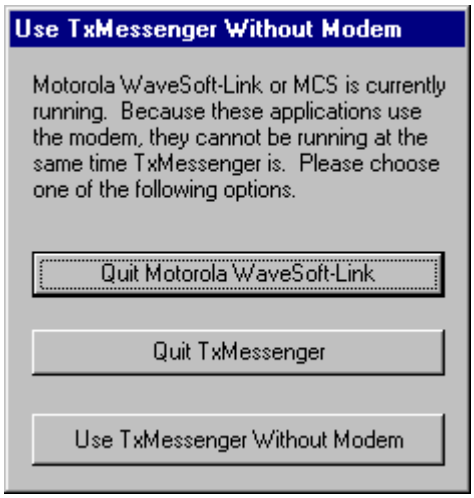
Another advantage to having TxMessenger and WaveSoft-Link installed on the same computer is that users can switch between using either program during the transition period. This is especially convenient for users that share a device, where not all the users have yet been trained to use TxMessenger. Having both applications available also provides some additional safety, in case incompatibilities or configuration differences prevent operation of TxMessenger.

A disadvantage to allowing users access to both WaveSoft-Link and TxMessenger is that they may tend to use what is more familiar. Thus delaying or lengthening the transition time.

12.3 Running WaveSoft-Link and TxMessenger at the Same Time

If more than one communication application runs at the same time (and are configured to use the same communications port), the first application to connect to the radio modem will prevent the other from accessing the radio modem.

To alert the user of potential communications-port conflicts, TxMessenger displays a warning dialog if WaveSoft-Link or MCS is already running when TxMessenger is started. This dialog provides the user the choice of quitting WaveSoft-Link, quitting TxMessenger, or continuing to run both applications (Use TxMessenger Without Modem).



The warning dialog can be disabled by setting `CloseApplicationWithThisWindowTitle` to blank.

If the WaveSoft-Link window name has been customized (WSLINK.INI setting APP_TITLE_TEXT) then TxMessenger needs to be configured with the same name of WaveSoft-Link's window (TxMessenger.ini setting CloseApplicationWithThisWindowTitle). The TxMessenger setting will match any window name with the same starting characters. Therefore, "WaveSoft-Link" matches windows with "WaveSoft-Link 100" or "WaveSoft-Link 200 12:00PM".

Failure to provide TxMessenger with the name of WaveSoft-Link's window may result in TxMessenger not noticing that WaveSoft-Link is running. In addition, if the user then presses the Quit WaveSoft-Link button in TxMessenger, WaveSoft-Link's communication engine (MCS) may be exited without exiting WaveSoft-Link first, causing instability in WaveSoft-Link.

 Important: WaveSoft-Link won't be able to access the radio modem and won't display a warning if TxMessenger is already running. Therefore, users that want to use WaveSoft-Link must be trained to first exit TxMessenger.

12.4 Autoconfigure from Existing WaveSoft-Link Settings

TxMessenger can configure itself based on the existing configuration of Motorola WaveSoft-Link or Motorola TX. The automatic configuration process saves technician time by converting many settings and providing an initial configuration similar to the prior messaging application.

Keep in mind:

- Not all existing WaveSoft-Link or TX settings are converted. Some settings don't have appropriate equivalents in TxMessenger and some settings would result in the loss or degradation of the more modern product features.

- Many TxMessenger settings are not available in the older products, so these settings must be configured manually if changes are desired.

- WaveSoft-Link and TX keys and labels (status, coded, etc.) were designed for less powerful devices (no touchscreen or mouse, less memory, less disk space, smaller screens, and lack of color) and other archaic limitations. Rather than burdening the user with the old setup, key and label configurations should be designed anew.

- All settings and product features should be tested to determine if performance is equal to or better than the prior messaging application. Emergency and core features should be rigorously tested. Because of the wide variety of systems and configurations, autoconfigure should not be relied upon to exactly reproduce the prior messaging application.

- Autoconfiguration doesn't eliminate the need for an initial expert technical configuration. It is not a substitute for system technologist services; it just saves time.

12.4.1 Initiating Autoconfigure

When TxMessenger is started, it searches for existing WaveSoft-Link or TX settings files if:

- the TxMessenger.ini file is missing
or…
- the TxMessenger.ini file contains ConfigureAuto=TRUE

The factory installed TxMessenger.ini has ConfigureAuto=TRUE by default.

To create a completely fresh TxMessenger.ini file:

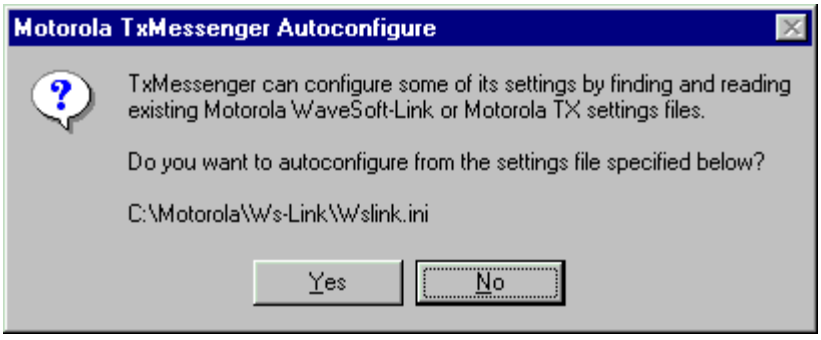
1. Exit TxMessenger.
2. Delete TxMessenger.ini.
3. Run TxMessenger.
4. Exit TxMessenger.

To merge a configured TxMessenger.ini file with existing WaveSoft-Link or TX settings:

1. Exit TxMessenger.
2. Set ConfigureAuto=TRUE in TxMessenger.ini.
3. Run TxMessenger.
4. Exit TxMessenger.

Any setting modified by autoconfigure will overwrite the previous value in TxMessenger.ini. In other words, existing WaveSoft-Link or TX settings (including key layouts) take precedence over the settings found in the original TxMessenger.ini.

If an existing WaveSoft-Link or TX settings file is found, a dialog is displayed:



12.4.2 Search Path for Motorola TX Settings Files

These are the files and the order in which TxMessenger searches for existing Motorola TX settings files:

```
\TxWin\TxWin.cfg
\TxWin\Labels.pfk
\Windows\Moto_sys.ini
```

If the settings files are found and the user chooses to autoconfigure from them, TxMessenger also copies and converts the forms located at:

```
\TxWin\Forms\
\TxWin\Queue\Forms\
```

If the settings files are not found or the user chooses not to configure from these files, TxMessenger searches its own directory for Motorola TX settings files:

```
\Program Files\Motorola TxMessenger\TxWin.cfg
\Program Files\Motorola TxMessenger\Labels.pfk
\Program Files\Motorola TxMessenger\Moto_sys.ini
```

12.4.3 Search Path for Motorola WaveSoft-Link Settings Files

These are the files and the order in which TxMessenger searches for existing Motorola WaveSoft-Link settings files:

```
\Motorola\Ws-Link\Wslink.ini
\Windows\Mcs.ini
```

If the settings files are found and the user chooses to autoconfigure from them, TxMessenger also copies and converts the forms located at:

```
\Motorola\Ws-Link\Rcvfmts\
\Motorola\Ws-Link\Rsdntfmt\
```

TxMessenger also copies and converts the help (*.CSH) files located at:

```
\Motorola\Ws-Link\
```

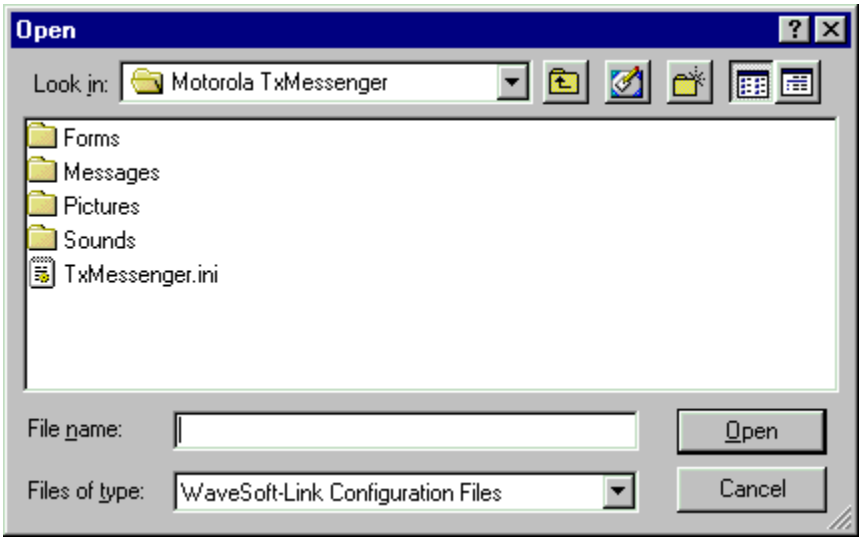
If the settings files are not found or the user chooses not to configure from these files, TxMessenger searches its own directory for Motorola WaveSoft-Link settings files:

```
\Program Files\Motorola TxMessenger\Wslink.ini
\Program Files\Motorola TxMessenger\Mcs.ini
```

12.4.4 Specifying a Path for a Settings File

Existing Motorola WaveSoft-Link or Motorola TX settings files in any directory can be merged with current TxMessenger.ini settings using the IMPORT command.

1. Run TxMessenger.
2. Open the Script Tester window (Ctrl-t, then click the Script Tester button)
3. Type IMPORT and press the Return key.
4. The following dialog appears:



From this dialog, select which settings file to convert.

Alternatively, the filename and path could have been specified after the IMPORT command. Such as `IMPORT "c:\Config\Wslink.ini"`.

12.5 TX, WaveSoft-Link, TxMessenger Comparison Table

	Motorola TX	Motorola WaveSoft-Link	Motorola TxMessenger
Operating System			
Windows 3.1x	Yes	Yes	
Windows 95		Yes	Yes
Windows 98			Yes
Windows NT 4			Yes
Windows 2000			Yes
Windows Me			Yes
Windows XP			Yes
Device			
Minimum memory	4MB	8MB	16MB
Minimum disk space available	4MB	10MB	16MB
Minimum display resolution	640x480	640x480	640x480
Movable, resizable window for different display sizes		Yes	Yes
Minimum display colors / grays	2	2	Thousands
Takes advantage of color displays?		Somewhat	Yes
Recommended minimum processor	80386	80486	Pentium 120mhz
Forte / Pen Windows-based devices	Yes	Yes	
MW-520 emergency button	Yes	Yes	Yes
MW-520 display buttons	Yes	Yes	Yes
On-screen labels for MW-520 display buttons			Yes
Touch-screen friendly		R01.03.00?	Yes
GPS			
Oncore GPS, Binary, NMEA, Loran	Yes		
External TAIP GPS	Yes	Yes	Yes
MW-520 Internal TAIP GPS		Yes	Yes
Network			
DataTAC 1.0 direct to modem	Yes	Yes	Yes
DataTAC 2.0 direct to modem	Yes	Yes	Yes
DataTAC 2.0 MWCS II FLM (“no” for all)			
DataTAC 2.0 MWCS II IP ("no" for all)			
ASTRO ("no" for all)			
GUI			
TX-like GUI	Yes		
WSL-like GUI		Yes	
TxMessenger-like GUI			Yes
Number of customer definable on-screen buttons	0	10	12
On-screen button order not always F1-F10 (For example: first on-screen button doesn't have to be F1)		WSL 200	Yes

On-screen buttons can be programmed to be different than F1-F10 which can be programmed to be different from MW-520s 6 display buttons	?	WSL 200	Yes
On-screen emergency button can have smaller click-area than other on-screen buttons.		Yes	
User-friendly forms screen	Somewhat	WSL 200	Yes
User-friendly status screen			Yes
User-friendly coded screen			Yes
User-friendly "other" (misc.) screen			Yes
Can launch other applications or documents			Yes
Keys screen	Yes	Yes	Yes
All keys on one screen		Yes	
Configurable start-up screen		WSL 200	Yes
Night mode		Yes	Yes
Configurable message and form colors		Yes	Yes
Multiple & on-the-fly message and form colors			Yes
Configurable window title bar text		Yes	
Title bar clock		Yes	
Command line		Yes	Yes
Command line history		Yes	Yes
Modem status	Yes	Yes	Yes
Modem information screen			Yes
New messages received indicator	?	Yes	Yes
On-screen save button (fixed or user-configurable)		Yes	Yes
On-screen scratch recall button (fixed or user-configurable)		Yes	Yes
Word wrap		Yes	Yes
Copy / paste		Yes	Yes
Tool tips			Yes
Blank screen when car in motion	9100s Yes		
Built-in screen saver	Yes		
Messages			
Status	Yes	Yes	Yes
Coded	Yes	Yes	Yes
Last received coded message displayed on main window	Yes	Yes	
Emergency	Yes	Yes	Yes
Plain text messages	Yes	Yes	Yes
Can print plain text messages on printer	Yes	Yes	Yes
Call Dispatch	Yes	?	Yes
Binary I/O port messages	9100s Yes		
Configurable inbound / outbound message priority		Yes	Yes
Reads TX message files	Yes		

Reads WSL message files		Yes	
Reads text message files			Yes
Received queue	Yes	Yes	Yes
Draft / scratch queue	Yes	Yes	Yes
Outgoing queue	Yes	Yes	Yes
Sent queue		Yes	Yes
Main window visual indication of priority messages	Yes		Yes
Main window visual indication of how many messages are in each queue	Yes	Yes	
Main window visual indication of how many unread messages there are	Yes	Yes	Yes
User can hide or display any of the queues at once	Yes	Yes	
LIFO/FIFO message sort order configurable	Yes	Yes	Yes
User can change message sort order on-the-fly			Yes
Doesn't accidentally display passwords in message queue	?		Yes
Can disable screen saver	Built-in Yes	Some Yes	Some Yes
Forms			
Long form titles	Yes	WSL 200	Yes
Auto suggests long form titles		WSL 200	Yes
Auto suggests associated pictures			Yes
Reads TX forms	Yes		Yes
Reads WSL forms		Yes	Yes
Reads text forms			Yes
Reads WSL Windows-like forms		WSL 200	
Windows-like forms (Resident or Protocol change)		WSL 200	
Customer create-able help page(s)		Yes	Yes
Customer create-able context sensitive help page for each form		Yes	Yes (script)
Tool tips for each form			Yes
Graphic characters	KDT		Yes
Images / Pictures			Yes
International			
International Latin-based forms and messages		Yes	Yes
International Latin-based interface text	Yes		Some customer modifiable now
Configuration			
Reads TX configuration file	Yes		Yes
Reads WSL INI file		Yes	Yes
Configurator Tool	Yes	Yes	Yes (built-in)
Auto detect modem mode, flow, and speed			
Host can remotely configure			Yes

Keys			
F1-F12 function keys	Yes	Yes	Yes
Shift F1-F12 function keys	Yes	Yes	Yes
Control F1-F12 function keys		Yes	Yes
Control-Shift F1-F12 function keys			Yes
MW-520 keys	Yes	Yes	Yes
Shift MW-520 keys			Yes
Control MW-520 keys			Yes
Control-Shift MW-520 keys			Yes
Emergency, status, coded, form, fill keys	Yes	Yes	Yes
Smart fill on Windows-like forms		WSL 200	
Key or on-screen button transmits current form if dirty	Yes	Yes	Yes
Dedicated transmit key			Yes
Navigate, special		Yes	Yes
Safety delay for emergency key		Yes	
Safety double-press for emergency button		R01.03.00?	Yes
Scriptable			Yes
Control key can be Forms key	9100s Yes		Yes
Sticky ALT/SHIFT/CTRL keys	Yes	Yes	Yes
Sticky keys indicators	Somewhat	Yes	Yes
Turns on caps-lock		buggy	
Caps lock indicator		Yes	
Uppercase only setting	Yes		Yes
Uppercase shift-lower case setting			Yes
Message Alerts			
Message alert fixed sounds	Yes	Yes	Yes
Message alert configurable sounds		WSL 200	Yes
Message alert pop-ups	Yes	Yes	Yes
Scripting			
Special functions		Yes	Yes
Multiple commands per key			Yes
Variables			Yes
Conditional statements			Yes
Branching			Yes
Start-up script			Yes
Shutdown script			Yes
API			
16-bit API	?	MDT Open	
32-bit API			Yes
MCS-compatible modem engine API (for PoliceWorks, etc.)		WSL 200	
GUI API			Yes

Logging		
Modem logging	Yes	Yes
GUI event logging		Yes
Installer		
Simple installer		Yes
Application rebuilds folders and INI if installation incomplete		Yes

12.6 Some Other Differences Between WaveSoft-Link and TxMessenger

12.6.1 WaveSoft-Link consists of two EXEs, a DLL, and system libraries (Pen Windows). TxMessenger.exe is the entire TxMessenger application. Because there is only one piece to TxMessenger, it is easy to determine the current version and impossible to mix up different pieces from different versions.

12.6.2 WaveSoft-Link has settings stored in different files, WSLINK.INI, MCS.INI, and NP_INIT.SYS. TxMessenger stores only modified settings, and those are all stored in TxMessenger.ini.

12.6.3 WaveSoft-Link's settings files can usually be altered while WaveSoft-Link is running, although some settings may get overwritten. TxMessenger's settings file can't be altered while TxMessenger is running, because all settings are overwritten upon exit. However, individual settings can be altered while TxMessenger is running by using the scripting language and Script Tester window, and those modifications are written back to the settings file.

12.6.4 WaveSoft-Link and MCS must be exited and rerun before a modified setting takes effect. TxMessenger's settings can be modified while running using the Script Tester dialog. Most of the setting changes take effect immediately, although some may require TxMessenger to be exited and rerun.

12.6.5 In forms with carriage returns within field definitions, WaveSoft-Link stops the field at the carriage return and continues the field at the beginning of the next line. TxMessenger can be configured to attempt to match the field length created by WaveSoft-Link, but TxMessenger displays the field contiguously.

12.6.6 WaveSoft-Link places a border around the current field, even on forms without field borders. TxMessenger does not draw a border around the current field.

12.6.7 WaveSoft-Link accepts downloaded text and fill keys from the host using type 4 forms (labels) name "TS" and "XS". TxMessenger always ignores these labels because TxMessenger.ini has a simplified (direct) settings format. Hosts can remotely configure TxMessenger using the type 4 form "CS". If `DownloadedFormNameCanReplaceExisting=TRUE`, then "TS" text labels are used to override form titles.

13 Key and Onscreen Button Configuration

There are four major areas in TxMessenger that are designed for user-activated customer-definable actions.

- **Standard Keyboard:** The F1-F12 function keys, the Escape key, and the Control key on a standard Windows keyboard.

- **Special Hardware Keyboard:** The six, white display buttons on the center-bottom of the MW-520 display. Also, any other hardware-specific keys that generate a Windows virtual keycode.

- **Onscreen Buttons:** Clickable buttons (either one or two rows) that appear at the center-bottom of the TxMessenger window

- **Other Screen Buttons:** Clickable picture buttons that appear in the center of the screen when the Other button is pressed.

The first two areas are invoked with keystrokes; the last two areas are invoked with mouse clicks (or touch-screen presses).

Because all four areas are independently scripted, all four areas can produce completely different actions from the other. For example, function key 1 doesn't have to be scripted to do the same thing as onscreen button 1, or MW-520 key 1, or any button in the Other screen. Of course, if desired, the exact same script can be copied into each configuration so that they do produce the same action and appear to the user to have some relationship.

Notice that the left-side menu buttons ("Desk" through "Transmit") aren't listed as customer-definable actions. That's because the left-side buttons have fixed functionality. Although access is provided to the script settings so that the buttons can be tweaked or patched, the action is not customer definable.

13.1 Settings for Keys

There are numerous types of keys that can be configured and labeled in TxMessenger.

All key settings have the same naming conventions:

The Script setting defines the action to be performed when the key is pressed (see the Script Command Reference in section 7 of this document).

This setting may be missing or blank, but then the key may not do anything when pressed.

```
KeyF1Script=ALERT "Example key pressed!"
```

The Help setting describes the text to display to the user in the Keys screen.

This setting may be missing or blank, but then no help text would appear for the particular key in the Keys screen.

```
KeyF1Help=Press me to display an alert.
```

The Label setting (if any) describes the name of the key to display to the user in the Keys screen. Except for virtual keys, it is recommended to not include key labels in TxMessenger.ini (thus leaving the internal default values intact) unless configuring for a foreign language.

```
KeysFunctionLabel=F^1
```

13.1.1 Function Keys

Function key scripts activate when the user presses a function key on the standard Windows keyboard. (# is from 1 to 12)

For F1 through F12:

```
KeyF#Script=
KeyF#Help=

KeysFunctionLabel=
```

For Control-F1 through Control-F12:

```
KeyF#ControlScript=
KeyF#ControlHelp=

KeysFunctionControlLabel=
```

For Shift-F1 through Shift-F12:

```
KeyF#ShiftScript=
KeyF#ShiftHelp=

KeysFunctionShiftLabel=
```

For Control-Shift-F1 through Control-Shift-F12:

```
KeyF#ControlShiftScript=
KeyF#ControlShiftHelp=

KeysFunctionControlShiftLabel=
```

13.1.2 Motorola MW-520 Display Key

MW520 key scripts activate when the user presses one of the white keys on the center-bottom of the Motorola MW-520 display panel. (# is from 1 to 6)

For MW520 key 1 through 6:

```
KeyMW520#Script=
KeyMW520#Help=

KeysMW520Label=
```

For MW520 key 1 through 6 while pressing the Control key on the standard keyboard:

```
KeyMW520#ControlScript=
KeyMW520#ControlHelp=

KeysMW520ControlLabel=
```

For MW520 key 1 through 6 while pressing the Shift key on the standard keyboard:

```
KeyMW520#ShiftScript=
KeyMW520#ShiftHelp=

KeysMW520ShiftLabel=
```

For MW520 key 1 through 6 while pressing the Control and Shift key on the standard keyboard:

```
KeyMW520#ControlShiftScript=
KeyMW520#ControlShiftHelp=

KeysMW520ControlShiftLabel=
```

13.1.3 Emergency Key

 **Important:** The customer and users themselves must test all combinations of the emergency button and emergency scripts in advance before relying on them in a crisis.

If the following setting is `TRUE`, an emergency message transmits when a script containing `TRANSMIT EMERGENCY` is executed. If the setting is `FALSE`, all (standard TX short-fixed message) emergency transmissions are disabled.

```
EmergencyTransmitAllowed=
```

If the following setting is greater than 0, two executions of a `TRANSMIT EMERGENCY` script are required within the specified time period (1 second = 1000 milliseconds) before emergency message transmission occurs. This helps prevent accidental activation.

```
EmergencyRequiresSecondActivationWithinThisManyMilliseconds=
```

If the following setting is TRUE, the transmission of an emergency message changes the first section of the Status Bar (for example, “TRANSMITTING”) and the icon in the Transmit button (for example, to a picture of transmission waves) just as any other transmitted message does. If FALSE, transmission of an emergency message is covert.

```
EmergencyTransmitShown=
```

13.1.3.1 Emergency Button on MW-520 Display

 **Important:** Unless some other application is providing emergency services, the emergency button doesn't do anything when TxMessenger isn't running. No emergency message is sent!

(Configuration not available in versions prior to v1.2)

By default, the MW-520 emergency button is scripted to transmit an emergency message when pressed. The action performed by the MW-520 emergency button can be altered without affecting other emergency transmission methods.

To disable the MW-520 emergency button, set the script to blank:

```
KeyMW5200Script=
```

To add functions to the MW-520 emergency button, make sure to include TRANSMIT EMERGENCY in the script if emergency transmission is still desired. Note that only the emergency transmission is performed covertly; other portions of the script are as apparent as if any other button or key had executed them.

```
KeyMW5200Script=TRANSMIT STATUS 7;TRANSMIT
EMERGENCY;
```

The MW-520 emergency button is normally listed on the Keys screen. It can be removed from the screen by setting the label to blank:

```
KeyMW5200Label=
```

Or the button can be "renamed" on the Keys screen:

```
KeyMW5200Label=Red Power Button
```

The description of what the button does can also be modified:

```
KeyMW5200Help=Keeps the computer from
turning off.
```

The emergency button can be scripted to perform different actions when pressed in conjunction with shift, control, and control-shift combinations. If no portion of a shift or control setting is defined, the emergency button executes the no-shift, no-control script.

```
KeyMW5200ShiftScript=
KeyMW5200ControlScript=
KeyMW5200ControlShiftScript=
```

The shift/control combinations of the emergency button can also appear or be hidden in the Keys screen, just like the plain emergency button.

13.1.4 Virtual Keys

Virtual key scripts activate when the user presses a key on the standard Windows keyboard. These settings allow almost any key on the keyboard to be scripted. (# is the keycode from 0 to 255)

```
Key#Script=
Key#Help=
```

Because the keycode number doesn't adequately communicate the user's perceived label of a key, the following label setting should be set to display the corresponding key press in the Keys screen.

```
Key#Label=
```

In practice, virtual keys are rarely used because the function keys are usually adequate. The notable exceptions are as follows:

13.1.4.1 Escape Key

The standard Windows keyboard Escape key is configurable through the following virtual key settings:

Key27Script=
Key27Help=
Key27Label=

13.1.4.2 Pause Key

The standard Windows keyboard Pause key is configurable through the following virtual key settings:

```
Key19Script=
Key19Help=
Key19Label=
```

13.1.4.3 Other Virtual Keys

The virtual keycodes generated by a particular keyboard can be displayed on the third section of the Status Bar with the following setting:

```
StatusBarKeyCodeLabel=^3
```

13.1.5 Sticky Keys

Normally, the user must press the Control, Alt, or Shift key at the same time as the key they wish to modify. For example: Press and hold Shift while pressing ‘a’ to obtain ‘A’.

To facilitate single-handed (or one-finger) operation, TxMessenger allows the Control, Alt, or Shift keys to stick on for a few seconds to modify the next key press. For example: Press and release Shift. Press and release ‘a’ to obtain ‘A’.

The following setting determines the number of thousandths of second (1 second = 1000 milliseconds) that the Control, Alt, or Shift keys will remain stuck on. Set to 0 or -1 to not allow any keys to stick.

```
KeysStickyEnabledMilliseconds=
```

Set the following setting to TRUE to allow the Control, Alt, and Shift key to be stuck on all at the same time. Example: Press and release Control. Press and release Shift. Both Control and Shift are on. Set to FALSE to cause the last key pressed to unstick the prior keys. Example: Press and release Control. Press and release Shift. Only Shift is on.

```
KeysStickySimultaneous=
```

Set the following setting to TRUE to allow a second press of the same sticky key to unstick. Example: Press and release Shift. Shift is on. Press and release Shift again. Shift is off. Set to FALSE to keep a key stuck on until another key press or until time has run out. Example: Press and release Shift. Shift is on. Press and release Shift again. Shift is still on.

```
KeysStickyToggles=
```

13.1.6 Control Key

A script can activate when the user presses the Control key on the standard Windows keyboard. However, if this script setting exists, the control key won't be available to modify other keys, like Ctrl-t or Ctrl-F1, nor will it activate as a sticky key.

```
KeyControlScript=
KeyControlHelp=
```

To emulate the next-form navigating ability of the "FORMS-Ctrl" key on the Motorola 9100 devices, set `KeyControlScript=SHOW NEXT UNREAD IN "FORMS"`

The following setting is the text displayed as the name of the Control key in the Keys screen. "Ctrl" by default.

```
KeyControlLabel=
```

The following setting is the text displayed in the third section of the Status Bar when the Control key is pressed and sticky keys are active. "Ctrl" by default.

```
KeysStickyControlLabel=
```

13.1.7 Alt Key

The following setting is the text displayed in the Keys screen for any key that also requires the Alt key to be pressed to activate. “Alt +” by default.

```
KeysAltLabel=
```

The following setting is the text displayed in the Keys screen for the name of when the user presses the Left Arrow and Alt key on the keyboard. “Alt + Left Arrow” by default.

```
KeysAltLeftLabel=
```

The following setting is the text displayed in the Keys screen for the name of when the user presses the Right Arrow and Alt key on the keyboard. “Alt + Right Arrow” by default.

```
KeysAltRightLabel=
```

The following setting is the text displayed in the third section of the Status Bar when the Alt key is pressed and sticky keys are active. “Alt” by default.

```
KeysStickyAltLabel=
```

13.1.8 Shift Key

The following setting is the text displayed in the third section of the Status Bar when the Shift key is pressed and sticky keys are active. “Shift” by default.

KeysStickyShiftLabel=

13.1.9 Capital Letters

The following setting determines the upper case or lower case state displayed for an alphabetic letter when typed.

`KeysCapitalLetters=`

If set to `Always`, the alphabetic letter is always converted to upper case, regardless of the Shift key or Caps Lock key. This is similar to having the Caps Lock key always enabled for forms and messages, except that the user can’t use the shift key to produce lower case letters.

If set to `On`, the alphabetic letter is always converted to upper case, unless the Shift key or Caps Lock key is pressed. In essence, upper case letters from the keyboard are made lowercase, and lower case letters are made upper case. This is equivalent to having the Caps Lock key always enabled for forms and messages.

If set to `Normal`, the alphabetic letter is accepted as is.

In all cases, the formatting sequence of a field always overrides the case state of a letter. If the field is auto-upper, the letters will always be upper case.

TxMessenger doesn’t actually alter the state of the Caps Lock key. For example, the user can switch to a report writing program with the Caps Lock off (for typing upper and lower case) and can then switch back to TxMessenger and type into a form or message as if the Caps Lock were on.

13.1.10 Insert Key

Depending on whether TxMessenger is in insert mode or overstrike mode, the text within a form or message on the Desk can be moved over or replaced by the characters being typed.

If the following setting is TRUE, then insert mode is enabled and characters are moved over to make room for any new characters that are typed. If FALSE, then overstrike mode is enabled and characters are replaced or “typed over”.

```
KeysInsertCharactersRatherThanOverstrike=
```

If set to `TRUE`, the following setting switches between insert mode and overstrike mode when the Insert key is pressed.

```
KeysInsertTogglesBetweenInsertAndOverstrike=
```

With the combination of these settings, the text mode can be:

Locked into overstrike mode:

```
KeysInsertTogglesBetweenInsertAndOverstrike=FALSE
KeysInsertCharactersRatherThanOverstrike=TRUE
```

Locked into insert mode:

```
KeysInsertTogglesBetweenInsertAndOverstrike=FALSE
KeysInsertCharactersRatherThanOverstrike=FALSE
```

Toggled by the user pressing the insert key:

```
KeysInsertTogglesBetweenInsertAndOverstrike=TRUE
```

13.2 Example Key Scripts

The following are only examples of some scripts that can be configured. See the Script Command Reference in section 7 of this document for a complete list and explanation of script commands.

13.2.1 Unassigned Key Example

Not defining a key script will cause it to perform the script defined by the setting `KeysOrButtonScriptIfBlank`. If this setting is set to blank, then pressing the key will do nothing.

With these settings, pressing the F1 key does nothing:

```
KeyF1Script=
KeysOrButtonScriptIfBlank=
```

With these settings, pressing the F1 key transmits the contents of the desktop:

```
KeyF1Script=
KeysOrButtonScriptIfBlank=TRANSMIT
```

13.2.2 Emergency Key Example

To define a key as an emergency button, set the setting to `TRANSMIT EMERGENCY`.

With this setting, an emergency message will be sent when Control + F1 is pressed:

```
KeyF1ControlScript=TRANSMIT EMERGENCY
```

13.2.3 Form Display Key Example

To display a specific form on the screen, set the setting to `SHOW FORM "formname"`. Formname can be a two-letter internal TX form name (like `"TA"`) or the actual filename (like `"Stolen Vehicle.txt"`).

With this setting, form TD will be shown when Shift + F1 is pressed:

```
KeyF1ShiftScript=SHOW FORM "TD"
```

13.2.3.1 Downloaded Form Keys / Labels Ignored

The TX-Protocol describes a method for downloading text keys / labels from the host. Because TxMessenger doesn't use an indirect or indexed setting organization for specifying keys, type 4 "TS" labels are ignored. If `DownloadedFormNameCanReplaceExisting=TRUE`, then "TS" text labels are used to override form titles.

Hosts can remotely configure keys or buttons for showing forms by using the remote scripting capability described in section 15 of this document.

Hosts can specify friendly form titles by making the first text line of the form reflect the form title. Or, the form title can be explicitly defined with a form title formatting sequence.

13.2.4 Download All Forms Key Example

To download all forms (on most systems), set the setting to `TRANSMIT STATUS 0`.

With this setting, a status #0 is sent and all forms will be downloaded when Control + Shift + F1 is pressed:

```
KeyF1ControlShiftScript=TRANSMIT STATUS 0
```

Considerable air time can be saved at shift changes by not sending a status 0 on startup (`StatusToSendAtStartup=-1`) and, instead, scripting a rarely used key (or Other button) to download all forms only when the user has been instructed that the forms have really changed. Note, however, that some hosts don't download forms when receiving a status 0, and some hosts require the status 0 to always be sent at the start of the shift for reasons in addition to form downloading.

13.2.5 Status Key Example

To send a status number, set the setting to `TRANSMIT STATUS number`, where the number is a value from 0 to 255.

With this setting, a status #1 is sent when the first MW-520 key is pressed:

```
KeyMW5201Script=TRANSMIT STATUS 1
```

13.2.6 Shutdown Status Key Example

To send a shutdown status, set the setting to `TRANSMIT STATUS 255`.

With this setting, the shutdown status is sent when Control and the first MW-520 key is pressed:

```
KeyMW5201ControlScript=TRANSMIT STATUS 255
```

13.2.7 Coded Message Key Example

To send a coded message, set the setting to `TRANSMIT CODED number`, where the number is a value from 0 to 255.

With this setting, a coded message #1 is sent when Shift and the last MW-520 key is pressed:

```
KeyMW5206ShiftScript=TRANSMIT CODED 1
```

13.2.8 Fill Key Example

To automatically fill a field with text, set the setting to TYPE "text".

With this setting, the text “BADGE #12345” will be automatically typed into the current field when the F10 key is pressed:

```
KeyF10Script=TYPE "BADGE #12345"
```

The Type command places the text into the keyboard queue to be read back as though the user typed it. The characters are modified by the Shift, Alt, and Control keys as they are read from the keyboard. This means a MW-520 or Function key that is activated in combination with another key (like Shift-F10 or Control-MW-520 #1) will have the modifier keys applied to the text. The text in the next example may be shifted upper or lower based on how quickly the Shift key is released to invoke Shift-F10:

```
KeyShiftF10Script=TYPE "Upper or lower case?"
```

13.2.8.1 Downloaded Fill Key Labels Ignored

The TX-Protocol describes a method for downloading fill keys from the host. Because TxMessenger doesn’t use an indirect or indexed setting organization for specifying keys, type 4 “XS” labels are ignored.

Hosts can remotely configure keys or buttons for filling by using the remote scripting capability described in section 15 of this document.

13.2.9 Multi-Function Key Examples

In TxMessenger, scripts aren't limited to only a single command.

With this setting, the fabricated vehicle inquiry form ('TT') is displayed, several fields are set to a default value, and the text insertion point is moved to the plate field when the user presses Control-Shift-F3:

```
KeyControlShiftF3Script=SHOW FORM "TT";SET FIELD
5 TO "IL";SET FIELD 2 TO "TRUCK";POSITION CARET
FIELD 4;
```

With this setting, the user can logoff the system and quit TxMessenger by simply pressing F12:

```
KeyF12Script=SHOW FORM "Logoff.txt";
TRANSMIT;TRANSMIT STATUS 255;EXIT;
```

13.3 Onscreen Button Configurations

TxMessenger has the ability to display up to twelve onscreen buttons that can be configured.

One row of onscreen buttons are displayed if `ButtonOnscreenNumberOfButtonsShown=6` or if `ButtonOnscreenNumberOfButtonsShown=-1` but no settings are defined for onscreen buttons greater than 6.

Two rows of onscreen buttons are displayed if `ButtonOnscreenNumberOfButtonsShown`=12 or if `ButtonOnscreenNumberOfButtonsShown=-1` and any setting for an onscreen button number greater than 6 is defined.

TxMessenger versions 1.1 and earlier don't feature the `ButtonOnscreenNumberOfButtonsShown` setting and act as though the setting were `-1`.

In the examples that follow in this chapter, # is a number from 1 to 12 to indicate which onscreen button the setting affects.

13.3.1 Onscreen Button Label

The label setting defines the button's title, which is the text displayed on the button itself at the bottom of the TxMessenger window.

```
ButtonOnscreen#Label=Example
```

If the text is too long and is cut off (usually with an ellipsis "…"), use \n to move some text to the next line. Text that requires more than two lines of text will not be displayed past the second line.

```
ButtonOnscreen#Label=Vehicle\nInquiry
```

The user can be reminded as to a function key (or any key) that performs the same action as the onscreen button by added the key name to the label. The key should then be given the same script as the onscreen button (copy and paste the script in TxMessenger.ini). Note that any onscreen button can be given any key name in any order. So, the first onscreen button could be labeled F12.

```
ButtonOnscreen#Label=Enroute\nF5
```

13.3.2 Onscreen Button Script

The script setting defines the action to be performed when the button is clicked (see the Script Command Reference in section 7 of this document).

This setting may be missing or blank, but then the button won't do anything when clicked.

```
ButtonOnscreen#Script=ALERT "Example onscreen
button activated!"
```

13.3.3 Onscreen Button Help

The help setting describes the text to display when the cursor is over the button and Tool Tips are turned on (`ToolTipsActive=TRUE`).

This setting may be missing or blank, but then no help text would appear for the particular button.

```
ButtonOnscreen#Help=Click me to display an alert.
```

13.3.4 Onscreen Buttons Examples

The following settings configure the first onscreen button to show the fabricated Log On form. The F2 key has been configured with the same help and script. The first onscreen button has its label end in “F2” so that the user will know that pressing F2 is the same as clicking the first onscreen button.

Placing the cursor over the first onscreen button displays the Tool Tip “Show the Log On form.”. Displaying the Keys screen (in the Other screen) shows that F2 will “Show the Log On form” also.

```
ButtonOnscreen1Help=Show the Log On form.
ButtonOnscreen1Label=Log On\nF2
ButtonOnscreen1Script=SHOW FORM "TO"
KeyF2Help=Show the Log On form.
KeyF2Script=SHOW FORM "TO"
```

The following settings configure the fifth onscreen button to change the status to 8, which means “Traffic Stop” on the example system. Placing the cursor over the fifth button would show ‘Send the Traffic Stop status’. Activating the fifth button would send a status #8. No matching key has been configured in this example.

```
ButtonOnscreen5Help=Send the Traffic Stop status.
ButtonOnscreen5Label=Traffic\nStop
ButtonOnscreen5Script=TRANSMIT STATUS 8
```

To get the second row of onscreen buttons to appear (when `ButtonOnscreenNumberOfButtonsShown=-1`), simply configure an onscreen button greater than 6.

ButtonOnscreen8Help=Show the Coded screen.
ButtonOnscreen8Label=Coded
ButtonOnscreen8Script=SHOW "Coded"

13.4 Status Message Button Configurations

TxMessenger can send a status number from 0 to 255. A status button sends a status number according to its setting's number. The host and device must be synchronized, agreeing on the meaning of a status number.

13.4.1 Configuring a Status Message Button

Example:

```
ButtonStatus1Help=Send the En Route status.
ButtonStatus1Label=En Route
ButtonStatus1Picture=StatusEnRoute.bmp
```

With these settings, the Status screen has this icon:

Activating this icon by clicking on it or selecting it with the arrow keys and pressing Return transmits a status #1 message.

If a picture setting is not specified or the specified picture isn't found, TxMessenger may display a suggested picture *(feature not available in versions prior to v1.2)*. To override a suggested picture, add a picture setting (`ButtonStatusXPicture` where `X` is the status number) that specifies an existing picture name or `StatusDefault.bmp`.

13.4.2 Rearranging the Status Message Button Order

Although not normally specified, status buttons can have a sort order. See section 10.1 on Other Buttons for details.

```
ButtonStatus1SortOrder=
```

13.4.3 Modifying the Default Status Message Button Behavior

Status buttons can have a script. If the script setting is missing, just the status number is transmitted. If the script setting is specified (even if blank), then no status message is transmitted unless the `TRANSMIT STATUS number` (0-255) command is included in the script.

```
ButtonStatus1Script=TRANSMIT STATUS 1
```

The replacement of the status script is useful if a status is no longer used:

```
ButtonStatus1Script=ALERT "En Route is no longer
used. Use Onroute instead."
```

Or if a status suggests additional steps that should be performed. For this example, imagine Status 1 is now Log Off:

```
ButtonStatus1Script=TRANSMIT STATUS 1;EMPTY
TRASH;EXIT;
```


The script should always transmit the same status number as the setting number, in this case: `TRANSMIT STATUS 1` for ButtonStatus1Script. In the status screen, TxMessenger selects a button based on the last status message sent, not the last button pressed. For example, if `ButtonStatus1Script=TRANSMIT STATUS 2`, then button 2 is going to be selected, even though button 1 was pressed by the user.

13.4.4 Downloading Status Buttons from the Host

Status buttons can be configured remotely by the host using the TX-Protocol. A type 4 message with the form name “SS” can contain a list of status buttons.

TxMessenger configures status settings (including suggested pictures if applicable) in TxMessenger.ini to match the downloaded list.

To ignore downloaded status buttons:

```
DownloadedStatusCanReplaceExisting=FALSE
```

13.4.5 Uppercase Status Buttons from the Host

Because many hosts send status button labels that are all UPPERCASE, TxMessenger can detect this condition and automatically capitalize the first letter of each word and lowercase the rest of the letters.

To disable automatic case adjustments for coded and status buttons:

```
DownloadedLabelsMadePretty=FALSE
```

13.4.6 Extra or Unused Status Buttons from the Host

Hosts are supposed to leave a status label blank (all spaces) if there is no status associated with that number. Some hosts use other words, such as "UNUSED" to indicate unused statuses. This would result in TxMessenger creating status buttons with the title "UNUSED".

To ignore status labels with a specific label:

```
DownloadedStatusUnusedLabel="UNUSED"
```

(substitute the text to ignore)

13.4.7 Displaying an Unknown Status

(Configuration not available in versions prior to v1.2)

When a status message is received from the host, but no setting with that status number exists, TxMessenger displays the text defined by `StatusUnknownText`. If the text contains ^1, then the status number is substituted.

Examples:
In TxMessenger.ini, `StatusUnknownText=#^1` but `ButtonStatus24Label` doesn't exist.
If status message of 24 is received, then "#24" is displayed on the status button and (if appropriate) the second section of the status bar.

In TxMessenger.ini, `StatusUnknownText=` but `ButtonStatus7Label` doesn't exist.
If status message of 7 is received, then nothing is displayed on the status button and (if appropriate) the second section of the status bar.

13.5 Coded Message Button Configurations

TxMessenger can send a coded message number from 0 to 255. A coded message button sends a coded message number according to its setting's number. The host and device must be synchronized, agreeing on the meaning of a coded number.

13.5.1 Configuring a Coded Message Button

Example:

```
ButtonCoded8Help=Send the Hold On coded message.
ButtonCoded8Label=Hold On
ButtonCoded8Picture=CodedHoldOn.bmp
```

With these settings, the Coded Message screen has this icon:

Activating this icon by clicking on it or selecting it with the arrow keys and pressing Return sends a coded #8 message.

If a picture setting is not specified or the specified picture isn't found, TxMessenger may display a suggested picture *(feature not available in versions prior to v1.2)*. To override a suggested picture, add a picture setting (`ButtonCodedXPicture` where `X` is the coded message number) that specifies an existing picture name or `CodedDefault.bmp`.

13.5.2 Rearranging the Coded Message Button Order

Although not normally specified, coded buttons can have a sort order. See section 10.1 on Other Buttons for details.

```
ButtonCoded8SortOrder=
```

13.5.3 Modifying the Default Coded Message Button Behavior

Coded buttons can have a script. If no script setting exists, then just the coded number is transmitted. If a script setting is specified (even if blank), then no coded message is transmitted unless the `TRANSMIT CODED number` (0-255) command is included in the script.

```
ButtonCoded8Script=TRANSMIT CODED 8
```

The replacement of the coded script is useful if coded messages are no longer used and text messages should be sent instead:

```
ButtonCoded8Script=TRANSMIT "HOLD ON A MINUTE!"
```

Or if a coded message suggests additional steps that should be performed automatically:

```
ButtonCoded8Script=TRANSMIT CODED 8;TRANSMIT
STATUS 2;SHOW FORM "TS";
```

 The script should always transmit the same coded number as the setting number, in this case: `TRANSMIT CODED 8` for ButtonCoded8Script. In the coded screen, TxMessenger selects a button based on the last coded message sent, not the last button pressed. For example, if ButtonCoded8Script=TRANSMIT CODED 9, then button 9 is going to be selected, even though button 8 was pressed by the user.

13.5.4 Downloading Coded Buttons from the Host

Coded buttons can be configured remotely by the host using the TX-Protocol. A type 4 message with the form name “MS” can contain a list of coded buttons.

TxMessenger configures coded settings (including suggested pictures if applicable) in TxMessenger.ini to match the downloaded list.

To ignore downloaded coded buttons:

```
DownloadedCodedCanReplaceExisting=FALSE
```

13.5.5 Uppercase Coded Buttons from the Host

Because many hosts send coded button labels that are all UPPERCASE, TxMessenger can detect this condition and automatically capitalize the first letter of each word and lowercase the rest of the letters.

To disable automatic case adjustments for coded and status buttons:

```
DownloadedLabelsMadePretty=FALSE
```

13.5.6 Extra or Unused Coded Buttons from the Host

Hosts are supposed to leave a coded label blank (all spaces) if there is no coded message associated with that number. Some hosts use other words, such as “UNUSED” to indicate unused coded messages. This results in TxMessenger creating coded buttons with the title “UNUSED”.

To ignore coded labels with a specific label:

```
DownloadedCodedUnusedLabel="UNUSED"
```

(substitute the text to ignore)

13.5.7 Coded Message Zero

The TX-Protocol specifies that the first downloaded label for coded message is ignored.

To allow the host to define a coded button that sends coded message #0:

```
DownloadedCodedLabel0Valid=TRUE
```

14 Status Bar Configuration

The Status Bar displays in the center at the bottom of the TxMessenger window, just above the Onscreen Buttons. The Status Bar consists of three sections, each displaying various states of the application. The Status Bar should not be confused with the Status Button (which is used to transmit messages regarding the state of the user or vehicle).

14.1 Status Bar Height and Number of Characters Displayed

The Status Bar height is controlled by the `StatusBarHeight` setting.

The larger the number for `StatusBarHeight`, the greater the text's font size. Since the text size increases with the Status Bar's height, smaller heights (such as 25) can display more characters, because the font width is smaller.

14.2 Showing / Hiding the Status Bar

To hide the Status Bar:
`StatusBarHeight=0`

To show the Status Bar:
`StatusBarHeight=32`
Any value from 25 to 35 shows the Status Bar.

14.3 Changing the Color of the Status Bar

The following setting controls the color of the Status Bar:
`StatusBarColor=`

A numeric color value or color name can be used.

14.4 Changing the Background Color of the Status Bar

The following setting controls the background color of the text in the Status Bar:
`StatusBarBackgroundColor=`

A numeric color value or color name can be used.

14.5 Status Bar Section 1

The text displayed in the first section of the Status Bar describes the current state of the modem or transmission activity.

When the first section of the Status Bar displays a transmission (such as pressing the transmit key), the following setting determines the text displayed:
`StatusBarTransmitLabel=`

When the first section of the Status Bar displays receiving an acknowledgement of a transmission, the following setting determines the text displayed:
`StatusBarTransmitAckLabel=`

When the first section of the Status Bar displays receiving a negative acknowledgement of a transmission, the following setting determines the text displayed:
`StatusBarTransmitNakLabel=`

The color of the text is based on the color setting for event being displayed.

To override the text in the first section, set StatusBarSection1Text to anything other than blank.

```
StatusBarSection1Text=Example 1
```

The overridden text remains displayed until the setting is deleted or is set to blank.

`StatusBarSection1TextColor` sets the color of the overridden text.

14.6 Status Bar Section 2

The text displayed in the second section of the Status Bar describes the last Status Message sent or received. This is the same Status Message that is displayed in the Status Button.

The color of the text is based on the color setting for Status Message being displayed.

Most recent Status Message transmitting but not yet acknowledged or was negatively acknowledged by the host.
`ButtonMenu5NakColor=`

Most recent Status Message acknowledged by the host:
`ButtonMenu5AckColor=`

Host transmitted a Status Message to the mobile device:
`ButtonMenu5ReceivedColor=`

To override the text in the second section, set `StatusBarSection2Text` to anything other than blank.

```
StatusBarSection2Text=Second Example
```

The overridden text remains displayed until the setting is deleted or is set to blank.

`StatusBarSection2TextColor` sets the color of the overridden text.

14.7 Status Bar Section 3

The text displayed in the third section of the Status Bar describes the sticky or modifier key currently pressed. For example: Shift, Alt, or Ctrl.

The color of the text is based on the setting `StatusBarKeyTextColor`.

Section 3 can also display the ASCII value, scan code, and key code of the last keypress.

```
StatusBarKeyCodeLabel=^1 S^2 K^3
```

To override the text in the third section, set `StatusBarSection3Text` to anything other than blank.

```
StatusBarSection3Text=Three!
```

The overridden text remains displayed until the setting is deleted or is set to blank.

`StatusBarSection3TextColor` sets the color of the overridden text.

14.8 Displaying the Most Recent Transmission in Status Bar

Any section in the Status Bar can be overridden to show the last activity of the user. The trade-off of forcing text into the Status Bar is that the text and color won't automatically change in that section anymore.

In the following example, the second section of the Status Bar is configured to display the last status, coded, form name, or message transmitted.

```
ButtonStatus1Label=  Enroute

ButtonStatus1Script= TRANSMIT STATUS 1;
                     StatusBarSection2Text="Status: Enroute"

ButtonCoded1Label=   Hold On

ButtonCoded1Script=  TRANSMIT CODED 1;
                     StatusBarSection2Text="Coded: Hold On"

TransmitDeskScript=  IF DeskFormName EMPTY;
                          temporary=TRUE;
                     ELSE IF DeskFormName="  ";
                           /* two spaces */
                          temporary=TRUE;
                     ELSE;
                          temporary=FALSE;
                     ENDIF;

                     IF temporary=FALSE;
                          StatusBarSection2Text="Form: ";
                          APPEND StatusBarSection2Text WITH DeskFormName;
                     ELSE;
                          SET temporary TO DESK;
                          TRIM temporary;
                          StatusBarSection2Text="Text: ";
                          APPEND StatusBarSection2Text WITH temporary;
                     ENDIF;

                     NEW MESSAGE;
```

(Remember: condense scripts onto a single line when adding them to TxMessenger.ini or typing them into the Script Tester.)

When Status button 1 is pressed, "Status: EnRoute" is displayed in section 2 of the Status Bar. When Coded button 1 is pressed, "Coded: Hold On" is displayed. When a form is transmitted, the two-letter internal form name is displayed. When a text message is transmitted, the starting text in the message is displayed.

To make the above example work completely, all status buttons, coded buttons, onscreen buttons, and keys that transmit status or coded messages must be scripted to change the `StatusBarSection2Text` as well.

The `NEW MESSAGE` command is only necessary if the user desires the screen to be cleared after a transmission.

Note that the `TRIM` and `APPEND` commands aren't available in TxMessenger version 1.1 or earlier. A `SET` command could be substituted for `APPEND`, except the nice label would be overwritten. TRIM could be dropped completely, but leading spaces in the text message would be displayed.

 **Important:** `SET StatusBarSection2Text TO DESK` can result in password fields being displayed if they occur early enough in the form. In the above example, just the form name is displayed instead of form contents to avoid displaying any passwords.

15 Remote Scripting and Configuration

TxMessenger can receive and execute scripts received from the host, CAD, or mobile device. The scripts can modify settings, thereby allowing remote configuration.

15.1 Scripts from the Host

The TX-Protocol doesn't specify a form type for scripting. TxMessenger supplements the protocol by accepting type 4 messages with the form name "CS" as scripts. (Type 4 messages are normally used for downloading labels.)

All text following the standard 7-byte TX-Protocol VLM header is considered a script and is executed by TxMessenger.

```
TXCS<00><40><00>PLAY SOUND "Startup.wav";ALERT
"Example of remote scripting";
```

(Don't transmit the "<" ">" characters themselves; just the hexadecimal byte values within.)

The routing bits are presently ignored, but should be set to 00 so that future scripting may use the bits to signify something.

If the host is prone to sending garbage or control characters after the end of messages, end the script with `;//` so that the remaining characters are interpreted as a comment.

Form formatting sequences (such as End Field) are not acceptable in scripts.

15.2 Receiving Script Results

The results of a script are returned to the host in a TX-Protocol VLM message of type 4 with the form name “CR”. This allows the mobile device to be queried for configuration, operating system, modem, or version information.

```
TXCS<00><40><00>
RESPOND "ApplicationVersion=";
RESPOND ApplicationVersion;
RESPOND "\x0D";
RESPOND "ModemSoftwareVersion=";
IF ModemSoftwareVersion EXISTS;
     RESPOND ModemSoftwareVersion;
ENDIF;
RESPOND "\x0D";
RESPOND "MachineModel=";
RESPOND MachineModel;
```

The above script might result in the following response:

```
TXCR<00><40><00>
ApplicationVersion=1.1
ModemSoftwareVersion=R01.02.19
MachineModel=Motorola MW-520
```

15.3 Remote Configuration Using a Script

Scripts can perform actions, return results, and modify settings. By modifying settings, some or all configuration can be made to a device (or the entire fleet) from the host, without having to physically touch the device.

```
TXCS<00><40><00>
KeyF1Script="TRANSMIT STATUS 15";
KeyF1Help="Transmits the Enroute status";
Exit="Alert Unread";
IF OperatingSystem = "Windows NT";
     WordWrap=FALSE;
ELSE;
     WordWrap=TRUE;
ENDIF;
```

Notice that because the configuration is performed in scripting, `IF` commands (as well as other commands) can be used to allow the same script to set different devices in the fleet to different configurations.

15.4 Remote Modification of a Status Bar Section

If desired, any or all sections in the Status Bar can be set to a small text message with a short remote script.

```
TXCS<00><40><00>
StatusBarSection3Text="Call Dispatch";
```

15.5 Alternate Method of Remote Scripting

Type 4 forms are usually impossible to generate except from the host. This provides a certain amount of security. Unfortunately, because this method is not part of the original TX-Protocol the host needs to be modified to provide this capability.

For hosts that can't be modified or where car-to-car or CAD generated scripts are desirable, TxMessenger can be given a plain-message keyword that causes the remainder of the message to be executed as a script.

```
CarToCarScriptHeader=
```

When a plain-text type 2 message (with a form name of 0x0000 or " " [two spaces]) is received, TxMessenger compares the beginning of the message to the entire contents of the `CarToCarScriptHeader` setting. If they match exactly, the remainder of the message is executed like a script.

If `CarToCarScriptHeader` is blank, this feature is disabled.

Example:

TxMessenger.ini already configured to:

```
CarToCarScriptHeader=Remote Command:
```

TxMessenger receives the following message:

```
TX  <00><20><00>
Remote Command: EXIT IMMEDIATELY;
```

Then TxMessenger executes everything after the text "Remote Command:". In this example, TxMessenger would quit.

16 Transmit

16.1 Host Errors When Using the Dedicated Transmit Button

16.1.1 The Dedicated Transmit Button

The dedicated transmit button is a green button in the lower-left corner of the TxMessenger window. The button is labeled “Transmit”. This button always transmits the contents of the Desk, even if a different screen (such as Forms or Status) is displayed.

Neither Motorola TX nor Motorola WaveSoft-Link had a key or button dedicated to transmitting. Instead, a function key (or some other key) had to be used for initiating transmissions. TxMessenger can be configured to be backward compatible with an existing transmit key, or the key can be replaced with a new function and the user can rely on TxMessenger’s dedicated Transmit button.

16.1.2 Keycode

When the dedicated Transmit button is used to transmit, the seventh byte of the TX-Protocol header is set to 0x00. Most hosts ignore this byte, but some require a number from 0x01 to 0x18 (0 to 24 decimal) to indicate which function key the user pressed to initiate transmission.

16.1.3 Determining if the Host Requires a Specific Keycode

If transmissions initiated with the dedicated Transmit button are being ignored, are generating error messages, or aren't returning the expected inquiry results, then the system probably requires a specific keycode. If a specific function key has been assigned or set aside as the transmit key, that is a good indication the system is expecting that specific keycode on transmissions.

The most accurate way to determine the correct keycode is to use an existing device that is communicating correctly with the host. Using RNC logging, a PC protocol analyzer, (or the MCS log file in WaveSoft-Link), look for the seventh byte of the TX variable length, message header on an inbound message.

Example:

… (some bytes may precede) `54 58 20 20 00 20 08` …(some bytes may follow)
`T X [ 1;1Htext of message`

In the above example, the keycode is 0x08. The `54 58` portion (equivalent to ASCII 'TX') of the example message are bytes that always start a TX variable length message header. However, the `20 20 00 20` portion of the example message may vary depending on the kind of message transmitted and on other settings.

If a log or analyzer isn't available, the keycode can be determined experimentally by stepping through all 24 possibilities.

16.1.4 Configuring the Dedicated Transmit Button's Keycode

The transmit button's keycode can be configured to any value from 0x00 to 0xFF with the following setting:

```
ButtonMenu7Script=  temporary=TransmitKeyCode;
                    TransmitKeyCode=0x08;
                    TRANSMIT;
                    TransmitKeyCode=temporary;
```

`ButtonMenu7Script` is the script that is executed when the seventh menu button (left side of the TxMessenger window) is pressed. A quick visual count confirms that the seventh button down is the Transmit button.

Normally, the script contains just "`TRANSMIT`". Two commands have been added before the `TRANSMIT` command and one command after.

The first command, "`temporary=TransmitKeyCode`", saves the current value of the `TransmitKeyCode` setting. If this command were omitted, you wouldn't know the original value of `TransmitKeyCode` after you changed it.

The second command, "`TransmitKeyCode=0x08`", sets the keycode that should be used. In this example, the keycode is 8 in hexadecimal. It can be "`TransmitKeyCode=8`" if decimal notation is preferred.

The third command, "`TRANSMIT`", actually transmits the Desk contents.

The last command, "`TransmitKeyCode=temporary`", restores the original value of the `TransmitKeyCode` setting. If this command were omitted, all the other buttons or keys that transmit would be stuck with a keycode of 8.

`temporary` is often used as a holding area for the original value of a setting because `temporary` doesn't get saved to TxMessenger.ini. It's a memorable name that's perfect for short-term storage.

16.1.5 Disabling the Dedicated Transmit Button

In the unlikely event that the administrator doesn't want users using the dedicated transmit button (and some key is configured for transmitting), the dedicated transmit button can be disabled as follows:

```
ButtonMenu7Script=ALERT "Press the F10 key to transmit";
```

or...

```
ButtonMenu7Script=//Do nothing
```

The help should probably also be changed:

```
ButtonMenu7Help=The Transmit button should not be
used. Use F10 (or whatever) instead.
```

The default accelerator, Alt-t, should probably also be changed, so it doesn't show up in the keys list:

`ButtonMenu7Label=Transmit`
(instead of `ButtonMenu7Label=&Transmit`. The & indicates which letter should be underlined and act as the accelerator key.)

Or, the label could indicate which key should be used instead:

```
ButtonMenu7Label=Transmit (F10)
```

Or, the label could be removed altogether:

```
ButtonMenu7Label=
```

16.2 Keycodes for Function Keys or Onscreen Buttons

(Read about keycodes in the above section regarding the dedicated transmit key.)

All keys, buttons, or other scriptable elements can be configured with any transmission keycode desired. Transmission keycodes can be assigned out of order, can be made unique for each key, can be made to all be identical, or can be made to vary for a single key based on what is being transmitted.

16.2.1 All Transmissions Use the Same Keycode

If `TransmitKeyCode` is set to any value other than -1 in the settings file, and that setting isn't modified by any script, then that value is used as the seventh byte of the TX header for all transmissions.

For example:
```
TransmitKeyCode=0x02
```
All transmissions send keycode 2.

16.2.2 Transmit Keycode Specified in Individual Scripts

If the `TransmitKeyCode` setting is set to any value other than -1 within the script of a button or key, the keycode transmitted in the seventh byte of the TX header will be the value specified by `TransmitKeyCode`.

For example:
```
KeyF1Script=temporary=TransmitKeyCode;TransmitKeyCode
=0x05;TRANSMIT "Hello";TransmitKeyCode=temporary;
```
Transmits "Hello" with the keycode 5 when the F1 key is pressed. Note that the `TransmitKeyCode` setting is restored to its prior value as the script ends, so that other keypresses won't be forced to use keycode 5.

16.2.3 Transmit Keycode Determined From Transmitted Form Name

`TransmitKeyCode` must be -1.

If `TransmitKeyCodeShouldBeBasedOnForm=TRUE`, the keycode transmitted in the seventh byte of the TX header will be the alphabetic value of the form name of the form being transmitted. Where TA=1, TB=2, TC=3, etc.

For example:
```
KeyF1Script=TRANSMIT;
```
If the TD form is displayed on the desk, the contents of the TD form are transmitted with the keycode 4 (TD=4) when the F1 key is pressed.

16.2.4 Transmit Keycode Determined From Requested Form Name

(This feature not available in versions prior to v1.3)

`TransmitKeyCode` must be -1.
`TransmitKeyCodeShouldBeBasedOnForm` must be `FALSE`.
Either `FormDisplayTransmitsIfDeskNotClear` or `FormDisplayTransmitsIfFormNotFound` must be TRUE

If `TransmitKeyCodeShouldBeBasedOnFormRequest=TRUE`, the keycode transmitted in the seventh byte of the TX header will be the alphabetic value of the form name of the form being requested. Where TA=1, TB=2, TC=3, etc.

For example:
```
KeyF1Script=SHOW FORM "TC";
```
If the TD form is displayed on the desk, the contents of the TD form are transmitted with the keycode 3 (the form requested is TC, and TC=3) when the F1 key is pressed.

16.2.5 Transmit Keycode Automatically Based On Key

`TransmitKeyCode` must be -1.
`TransmitKeyCodeShouldBeBasedOnForm` must be `FALSE`.
`TransmitKeyCodeShouldBeBasedOnFormRequest` must be `FALSE`.

The keycode of the physical keyboard is used for transmission.

For example:
`KeyF1Script=TRANSMIT;`
Whatever is on the desk is transmitted with the keycode 1 when the F1 key is pressed.
`KeyF2Script=TRANSMIT;`
Whatever is on the desk is transmitted with the keycode 2 when the F2 key is pressed.
(TxMessenger v1.2 and earlier used a much larger value for function keys, but they still varied based on the actual key pressed.)

17 Right-Click Configuration

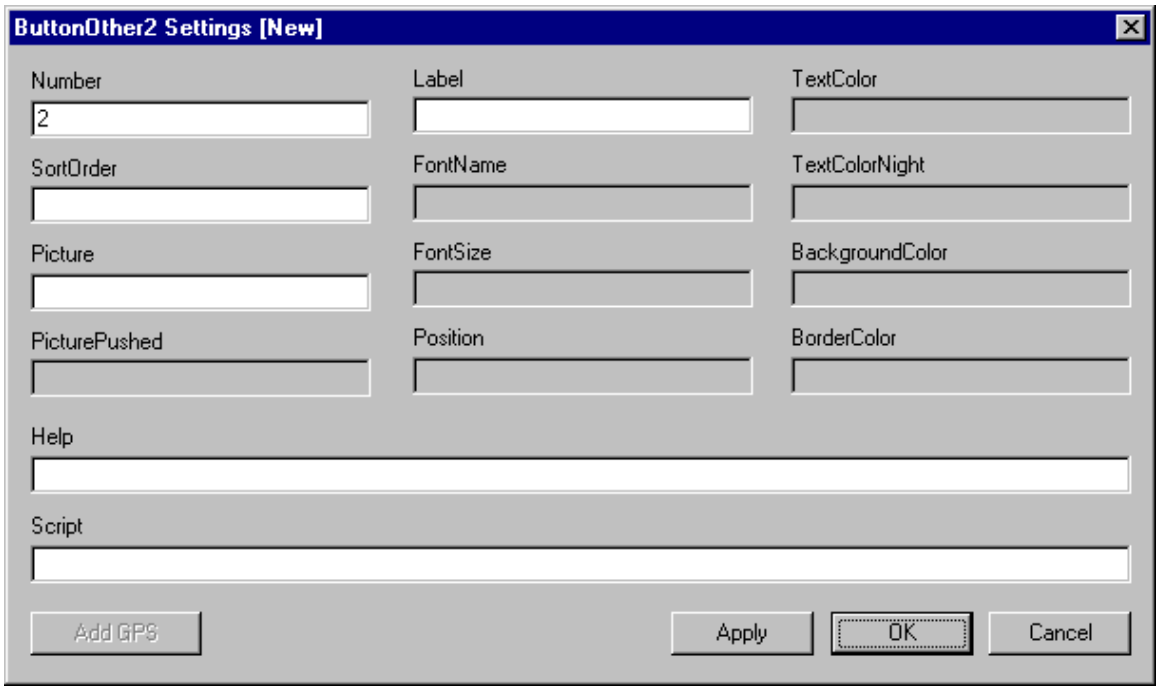
Beginning with TxMessenger version 1.3, right-clicking some screen items displays a menu. The choices available in the menu vary based on the screen item clicked as well as the version of TxMessenger.

Right-click configuration can be disabled by setting `Configure=FALSE`.

Beginning with TxMessenger version 2.0, some configuration dialogs will have an Add/Remove GPS button. This button will either add or remove text from the script for the item being configured. The text added or removed is defined by the setting `GPSRunTransmitScriptText` . By default, `GPSRunTransmitScriptText=Run GPSTransmitScript.` Adding this text to the beginning of a script will allow GPS data to be transmitted when the button is activated.

Note: This button is only enabled if the setting GPSLicenseNumber has been set to a valid license number.

17.1 New

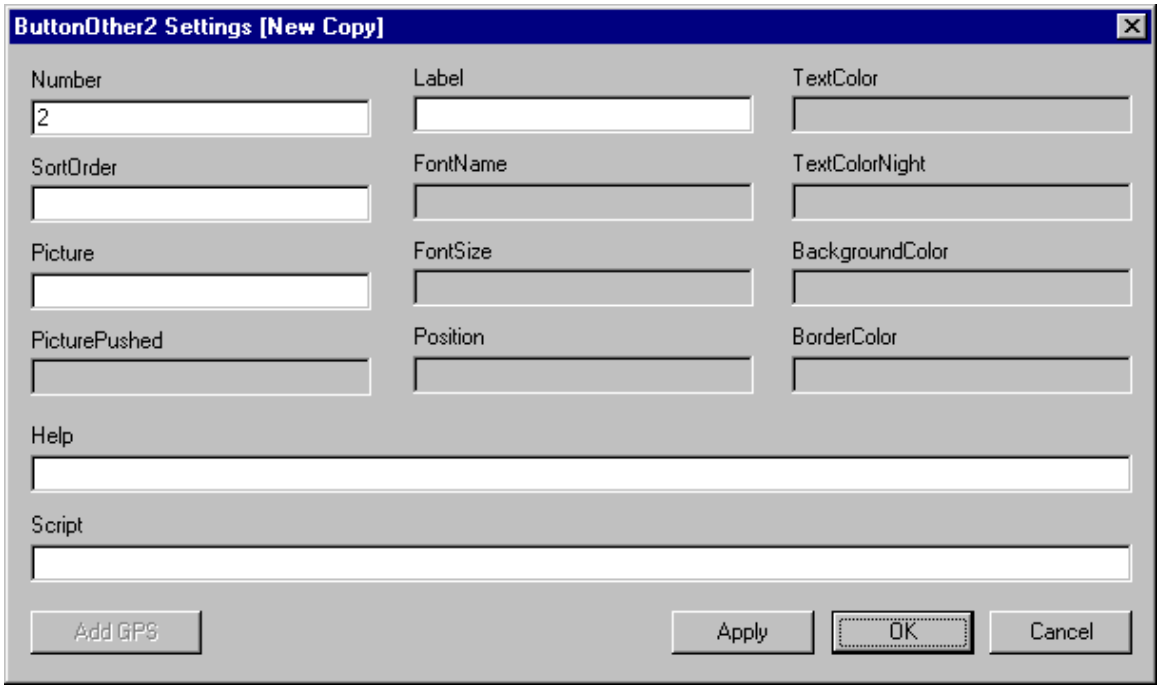


Right-click on no button (in the Extras, Forms, Status, Coded, Tools, or Other screen) and choose New to display the New settings dialog for that button. The name of the dialog is the name of the type of button found on the screen that was clicked. This dialog was shown when New was selected on the Other screen.

Each field is labeled with the name of the TxMessenger.ini setting less the button name. The field themselves contain the value of the setting.

The value in the Number field is the first available number for that particular type of button.

17.2 New (Copy)

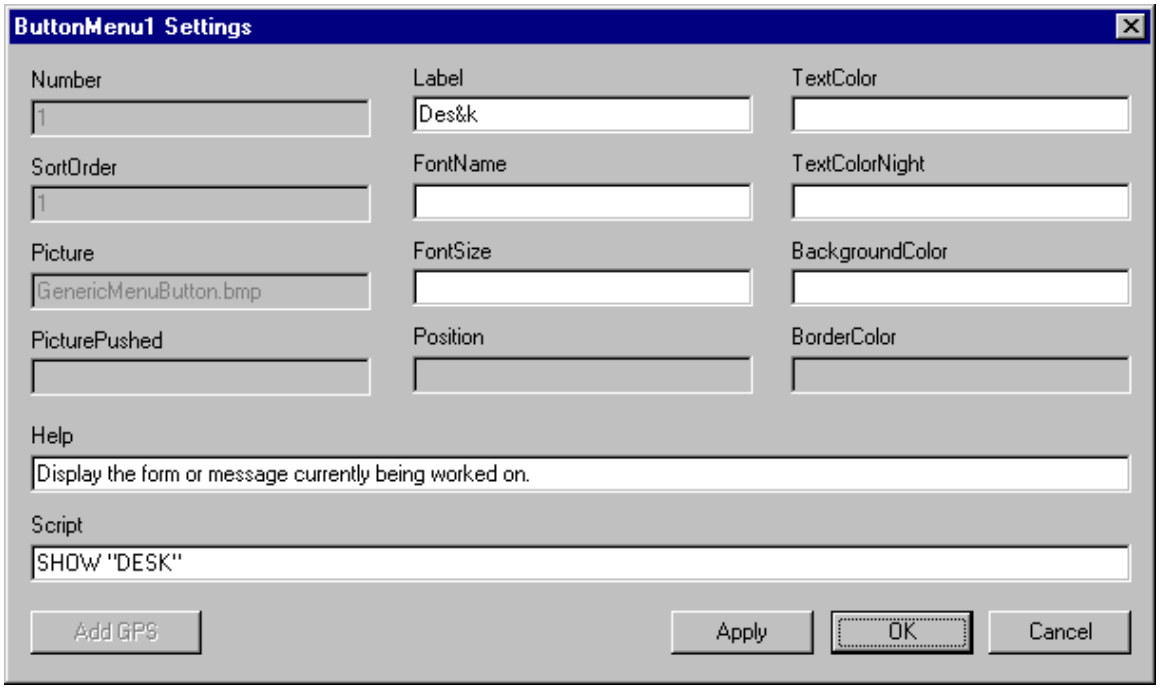


Right-click on a button (in the Extras, Forms, Status, Coded, Tools, or Other screen) and choose New (Copy) to display the New Copy settings dialog for that button. The name of the dialog is the name of the type of button found on the screen that was clicked.

Each field is labeled with the name of the TxMessenger.ini setting less the button name. The field themselves contain the value of the setting.

The value in the Number field is the first available number for that particular type of button.

17.3 Item Settings

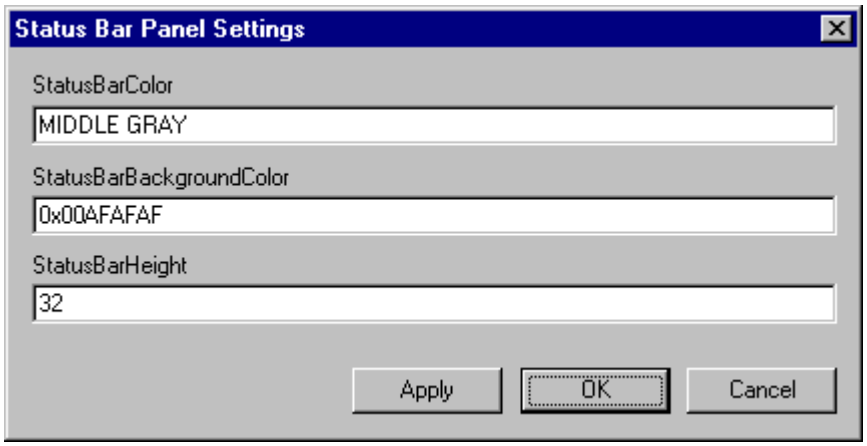


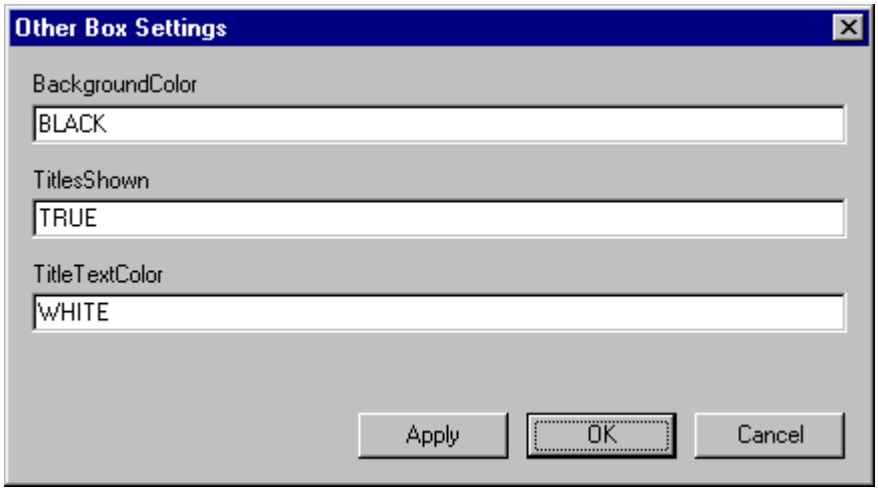
Right-click a button (on the main window or in the Extras, Forms, Status, Coded, Tools, or Other screens) and choose Item Settings to display the settings dialog for that button. The name of the dialog is the name of the button clicked.

Each field is labeled with the name of the TxMessenger.ini setting less the button name. The field themselves contain the value of the setting.

Some fields for some buttons may be disabled, as they are locked by the TxMessenger application. All fields in the Forms Item Settings screen are read-only, because those are derived from Formatting Sequences that are part of the forms themselves, not settings.

17.4 Box and Panel Settings



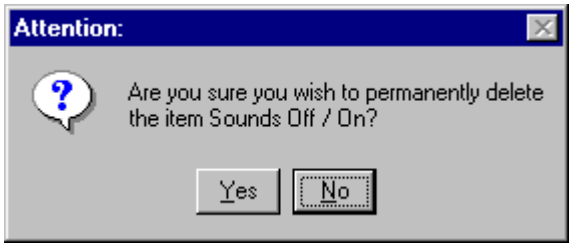


Right-click on or over a group of buttons (on the main window or in the Extra, Forms, Status, Coded, Tools, or Other screens) and choose Box or Panel Settings to display the settings dialog for that entire screen or group of buttons. The name of the dialog is the name of the box or panel which is being edited.

Each field is labeled with the name of the TxMessenger.ini setting less the box or panel name. The field themselves contain the value of the setting.

No Box and few Panel settings were available in versions prior to v2.5.

17.5 Delete



Right-click a button (on the onscreen buttons or the Forms, Status, Coded, Tools, or Other screens) and choose Delete to clear or remove the item clicked.

The name described in the confirmation dialog is the name of item clicked. Pay attention to the name, not the currently selected item (usually a square blue border around a form or status message), as right-clicking doesn't select an item.

Deleting most items simply clears all the settings associated with it. Deleting a form deletes the file from the Forms folder. Deleting an Other or Tool button places a blank label for that button in the settings file.

Important: Deleting an item is not undoable. Keep a backup copy of forms and TxMessenger.ini. However, the Other or Tool buttons can be restored by removing their associated blank label setting from TxMessenger.ini.

18 Printing

18.1 General

Beginning with version 1.5, TxMessenger can print documents (a form or a message). TxMessenger can manually print a document that is currently on the desktop or automatically print an arriving document that has the print bit set from the host.

TxMessenger printing is what-you-see-is-what-you-get (WYSIWYG). It does not do any formatting or arranging of the document. What you see on the desktop is what will be printed. *Exception*: TxMessenger always prints the document with a light background and dark text (day mode) regardless of the background color or current video mode.

An empty message (such as a clear desk) is not printed.

Important: Printing is enabled by default. If a printer is not connected or a default printer has not been selected, the user may see the following dialog from Windows:

18.1.1 Printer Compatibility

TxMessenger is not pre-certified to print on all printers because of the vast number of commercial printers that are available. It is the customer's responsibility to verify that TxMessenger can adequately print documents using the customer's existing or proposed hardware.

18.1.2 Disabling the Print Button

To hide the print button, set:

```
ButtonNavigationPrintShown=FALSE
```

The print button won't be shown on the Navigation bar with this setting value.

Note: Any changes made to the default Navigation button settings in TxMessenger versions prior to 1.5 need to be modified to reflect the new Navigation button numbering scheme. The Clear button is now numbered 2 instead of 1, the Delete button is now numbered 3 instead of 2, and so on.

18.1.3 Disabling the Automatic Print Bit

If the print bit is set on an incoming message or form, TxMessenger automatically prints the message or form.

The print bit can be forced off for specific form types or priorities by changing the routing bit setting(s) as desired:

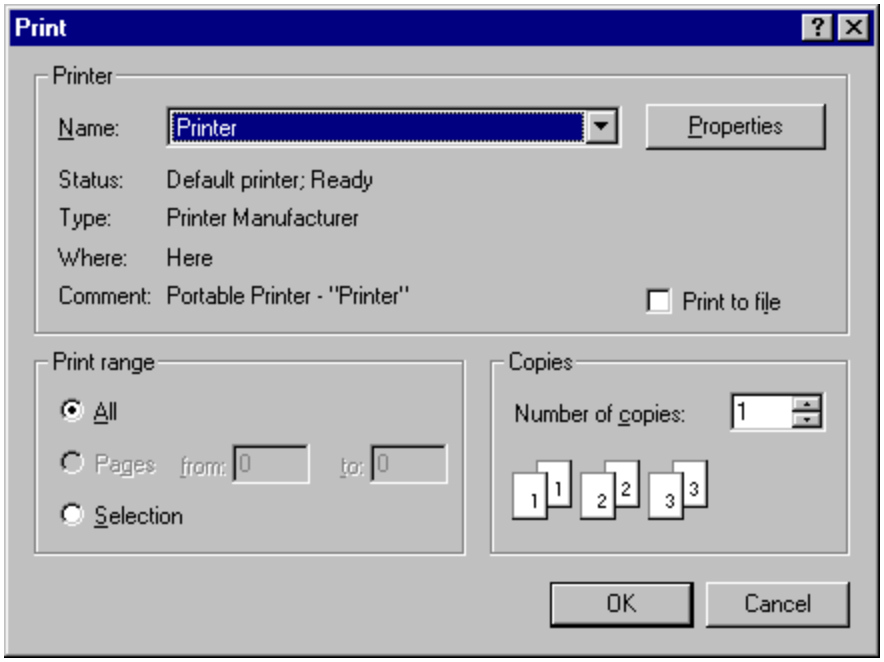
```
RoutingBitsANDFormType1=0b11011111
RoutingBitsANDFormType2=0b11011111
RoutingBitsANDFormType3=0b11011111
RoutingBitsANDFormType4=0b11011111

RoutingBitsANDPriority0=0b11011111
RoutingBitsANDPriority1=0b11011111
RoutingBitsANDPriority2=0b11011111
RoutingBitsANDPriority3=0b11011111
```

Notice all other bits pass through (`1`) except the third most-significant bit (`0`) which is turned off. The third most-significant bit is the print bit.

18.2 Print Dialog

A Print dialog can be displayed when the Print button is selected:



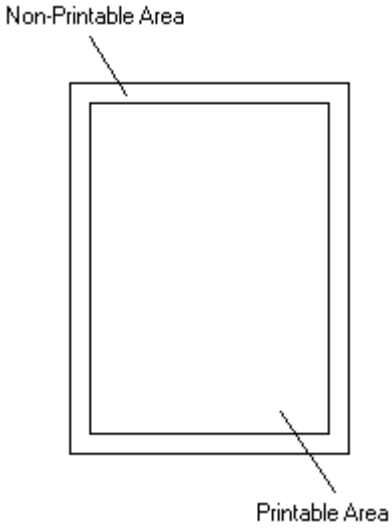
Printer options such as the target printer, print range, and number of copies can be selected within this dialog. After the desired options are set, click the OK button to print the document.

This dialog can be displayed when the Print button is pressed or the `PRINT` script command is executed:

```
PrintPropertiesDialogShown=TRUE
```

By default, this setting is set to FALSE. With the default setting, all documents are printed using the default printer and default print options.

18.3 Margins



TxMessenger prints the selected document by scaling it to fill the printable area of the selected page size.

This printable area can be adjusted by using the `PrintMarginBottom`, `PrintMarginLeft`, `PrintMarginRight`, and `PrintMarginTop` settings. By default, these settings are set to 0 so the entire printable area is used.

18.3.1 Scaling

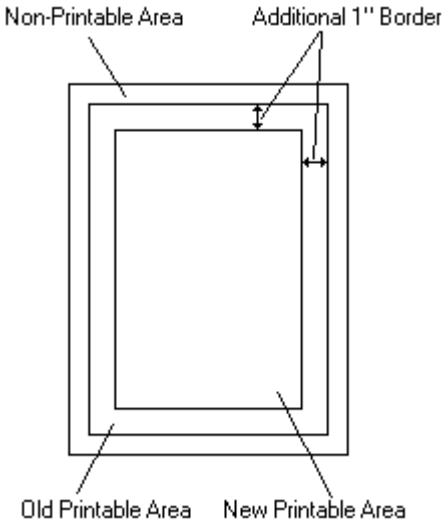
Because TxMessenger scales the document to fill the printable area, the `PrintMargin` settings can be used to shrink or expand the selected document.

Note: any portions of the document outside of the printable area will not be printed. Thus, the printable area is in effect the largest size the entire document can be.

For example, setting:

```
PrintMarginTop=1
PrintMarginBottom=1
PrintMarginLeft=1
PrintMarginRight=1
```

produces a printable area 2” smaller in width and height than the original printable area:



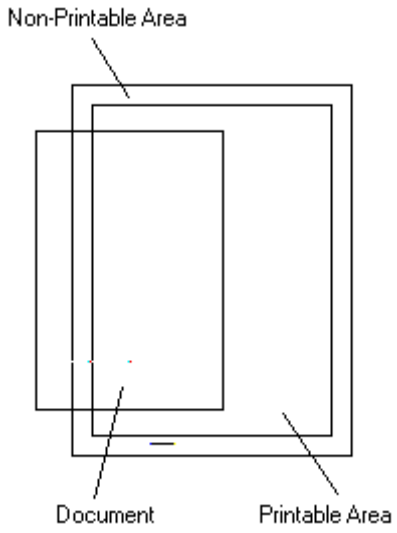
18.3.2 Positioning

A document can be positioned such that portions of it would be moved out of the printable area and not printed.

For example, setting:

```
PrintMarginLeft=-2
PrintMarginRight=2
```

creates this type of printout:



The document has been shifted to the left by 2 inches. This moves the left 2 inches of the document out of the printable area and therefore will not be printed.

19 TxMessenger API

19.1 General

Beginning with version 1.5, TxMessenger can provide copies of incoming or outgoing messages and forms to other applications. Applications can also send scripts to TxMessenger or receive text data via the Extension script command.

19.2 API Requirements

In order to communicate with TxMessenger, an application must be able to process the Windows `WM_COPYDATA` message.

A sample API application and source code is included on the TxMessenger media. The source code details the various data types defined to distinguish between the different types of data (scripts, messages, results, and so on) exchanged with TxMessenger.

19.3 Received Variable-Length Messages

An application can automatically receive copies of all received messages, forms, and labels (TX-protocol variable-length messages) as TxMessenger receives them. Existing forms and messages aren't provided, only new arrivals are.

TxMessenger doesn't alter or filter the messages. The raw data is simply passed along from TxMessenger's modem engine.

19.3.1 Receiving Received

To specify the receiving application, add the application's window class name or window title to the setting `ExtensionApplicationsReceive:`

```
ExtensionApplicationsReceive=Your Application Title
```

To specify more than one application, use a comma to separate window class names or titles:

```
ExtensionApplicationsReceive=First App, Second App
```

19.4 Transmitted Short-Fixed and Variable-Length Messages

An application can automatically receive copies of all transmitted forms, labels, status, coded, and emergency messages as TxMessenger transmits them. Existing sent messages aren't provided, only new transmissions are.

TxMessenger doesn't alter or filter the messages. The raw data is simply passed along from TxMessenger's modem engine.

19.4.1 Receiving Transmits

To specify the receiving application, add the application's window class name or window title to the setting `ExtensionApplicationsTransmit:`

```
ExtensionApplicationsTransmit=Your Application
```

To specify more than one application, use a comma to separate window class names or titles:

```
ExtensionApplicationsTransmit=First App, Second App
```

19.5 Sending Scripts to TxMessenger

An application can initiate a script in TxMessenger using the same commands and capabilities that are available to function keys. See the Script Command Reference (Chapter 7).

Scripts can be used for a wide variety of purposes, such as modifying settings, returning setting values, filling out fields on forms (like from a magnetic stripe reader program), changing status, or transmitting messages.

Applications can send scripts to TxMessenger by using the Windows `WM_COPYDATA` message and the `RECEIVED_SCRIPT` data type. The result of the script is returned. See the TxMessenger API Tester sample application for further details.

19.6 Sending Data or Commands from TxMessenger

TxMessenger can initiate a script or pass data to another application by using the script Extension command. The syntax is Extension “application window class or title” “Data to send”. Here is an example:

```
Extension "TxMessenger API Tester" "Hello"
```

This script command will send the text “Hello” to the TxMessenger API Tester, specifying that the type of data sent is `SENDING_DATA`.

Of course, a variable or setting can be used instead of `"Hello"`.

19.7 Interacting With Applications Without the API

Applications that don't support this API but do support DOS command parameters can be started up with variables.

```
SET launchcommand TO
"c:\\\\program files\\\\makemap\\\\makemap.exe ";

SET latitude TO FIELD 5;
TRIM latitude;
SET longitude TO FIELD 6;
TRIM longitude;
SET launchparameters TO latitude;
APPEND launchparameters WITH " ";
APPEND launchparameters WITH longitude;

SET temporary TO launchcommand;
APPEND temporary WITH " ";
APPEND temporary WITH launchparameters;
ALERT temporary;

START launchcommand launchparameters;

DELETE latitude;
DELETE longitude;
DELETE launchcommand;
DELETE launchparameters;
```

This script reads whatever is in field 5 and field 6 of the currently displayed form and displays an alert telling you the command it is about to execute. After you click OK, if there is actually a program at that location on the drive, it will be launched with those parameters.

You could even add an `IF` to the onscreen button that would read the value of `DeskFormName` and launch different programs or use different fields depending on which form is currently displayed.

The API adds the ability to pass these parameters to an application that is already running. Instead of `START`, you would be able to use `EXTENSION`. Also, the API means that the map program can sign up to get a copy of all messages as they arrive.

20 Command Line

20.1 General

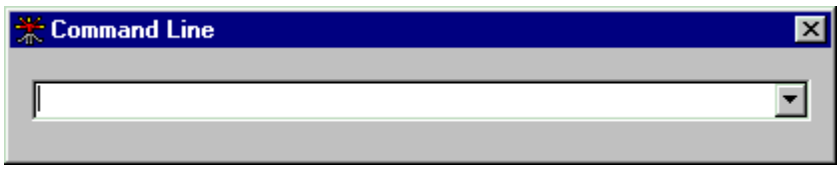
Beginning with version 1.6, TxMessenger has a command line interface that executes a user-defined script when the return key is pressed. The default script transmits the text typed in the Command Line dialog edit box.

20.2 Displaying the Command Line

To display the command line, execute the script:

```
SHOW "COMMAND";
```

This displays the dialog:



20.3 Using the Command Line

When the return key is pressed, the contents of the edit box are placed into a temporary variable named `CommandLineText`. For example, if "Hello" is typed into the Command Line and then the return key is pressed, the value of `CommandLineText=Hello`.

What happens next is determined by the value of `CommandLineScript`. By default, `CommandLineScript=Transmit CommandLineText`. In this situation, the text "Hello" is transmitted.

Another possible use for the Command Line is to execute scripts. If `CommandLineScript=Run CommandLineText`, then the text typed into the Command Line dialog would be executed as if it were a script. For example, typing in SHOW "FORMS" and hitting the return key would cause the Forms page to be displayed.

21 Copy And Paste

21.1 General

Beginning with version 1.6, TxMessenger has the capabilities of copying and pasting text. Text can be copied from forms on the desktop and pasted to other applications (such as Notepad) and vice-versa.

Note: Characters copied from password fields are turned into asterisks to protect the contents of the password field.

21.2 Selecting Text

21.2.1 Selecting Text using the Keyboard

To select text using the keyboard, hold down the shift key and press one of the directional arrow keys in the direction you wish to select. All text between the caret's initial position (where the shift key was first held down) to the position before the caret's current position (where the arrow key was released) becomes selected. Text can also be selected using the shift and end/home keys.

21.2.2 Selecting Text using the Mouse

21.2.2.1 Selecting by Dragging

To select text using the mouse, press and hold the left mouse button and drag the cursor over the text you wish to select. The text you may select depends on the value of the setting `EditSelectionStyle`.

`EditSelectionStyle` may be set to `All`, `Edit`, `Field`, or `Selective`.

`All` – There are no restrictions on the text the user can select. Both read-only and editable text may be selected.

`Edit` – Only editable text may be selected. The mouse can be dragged over the entire form and only editable text will be selected.

`Field` – Text selection is restricted to the editable text in the field that the mouse drag began. If the mouse drag begins on read-only text, nothing is selected.

`Selective` – Text selection restrictions depend on where the mouse drag begins. If the drag begins on read-only text, then both read-only and editable text can be selected. If the drag begins on editable text, then only editable-text can be selected.

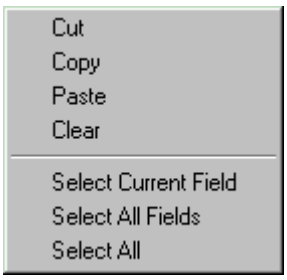
21.2.2.2 Selecting by Double Clicking

Double clicking on a word using the mouse will select that word. Double clicking on an empty field will select nothing.

21.3 Copy and Paste Commands

21.3.1 Using the Right-Click Menu

Clicking the right-mouse button over the desktop area displays a copy and paste menu:



Certain menu items may be grayed out if the command for that menu item is not valid at that moment. For example, if read-only text is selected, the Cut command will be grayed out because read-only text cannot be altered.

21.3.2 Using the Hot Keys

Clipboard keystrokes can be scripted to perform different commands or can be removed by setting their script values to blank. By default, the following keystrokes perform the following tasks:

Control + A: Selects all text (including read-only and fields).

The settings for this hot key are:

```
Key65ControlHelp=Selects all text (including read-only and
                 fields).
Key65ControlLabel=Control + A
Key65ControlScript=SELECT TEXT ALL
```

Control + C: Copies selected text to the clipboard.

The settings for this hot key are:

```
Key67ControlHelp= Copies selected text to the clipboard.
Key67ControlLabel=Control + C
Key67ControlScript=COPY
```

Control + V: Paste text from the clipboard where the caret currently is.

The settings for this hot key are:

```
Key86ControlHelp=Pastes text from the clipboard
                 starting at the current caret position.
Key86ControlLabel=Control + V
Key86ControlScript=PASTE
```

Control + X: Copies selected text to the clipboard before clearing it.

The settings for this hot key are:

```
Key88ControlHelp=Copies selected text to the clipboard
                  before clearing it.
Key88ControlLabel=Control + X
Key88ControlScript= COPY; CLEAR TEXT
```

21.3.3 Copy and Paste Script Commands

The following script commands are now available:

```
CLEAR TEXT
COPY
COPY "text"
PASTE
PASTE "text"
SELECT TEXT horizontal vertical
SELECT TEXT horizontal vertical TO horizontal vertical
SELECT TEXT ALL
SELECT TEXT ALL FIELDS
SELECT TEXT FIELD
SELECT TEXT FIELD "fieldName"
SELECT TEXT NONE
SET settingName TO CLIPBOARD
SET settingName TO SELECTION
```

See section 7.3, Commands, for details on what these scripts do.

22 GPS

22.1 General

Beginning with version 2.0, TxMessenger has the ability to send/receive ASCII data to/from any device connected to a serial communications port. Although any device can be connected to the serial port, this chapter focuses on how to connect to a Trimble 450 Placer GPS device.

GPS is enabled only if `GPSLicenseNumber` is set to a valid license number. See section 6.1.4 on the About TxMessenger screen for details on how to determine whether or not GPS has been enabled.

 **Important:** The GPS routines in TxMessenger are designed to generically send and receive ASCII data from a host to a serial device. TxMessenger doesn't have any knowledge regarding specific GPS devices, hosts, or any of the data being passed through the GPS channels. TxMessenger really acts as a messenger service and provides no processing. It is the integrator's responsibility to configure each element in the system properly.

 **Warning:** Failure to regularly read and remove data from the GPS device may cause the buffer to overflow (see GPSReadScript below). Under those circumstances, some GPS data may be lost and partial GPS results may be received and transmitted.

22.2 TxMessenger GPS Commands

The following commands have been added:

```
GPS READ
GPS WRITE
TRANSMIT GPS
```

See Chapter 7 for details on what these commands do.

22.3 Serial Communications Port Settings

In order to communicate properly with the device, the serial port must be configured correctly. The following settings can be used to configure the serial port:

```
GPSBaud
GPSDataBits
GPSPort
GPSStopBits
```

The settings required for the Trimble Placer 450 GPS are:

```
GPSBaud=9600
GPSDataBits=8
GPSStopBits=OneStopBit
```

`GPSPort` can be set to any available serial port. By default, `GPSDataBits` and `GPSStopBits` are set to the most commonly used values.

Note: The internal GPS device for the Motorola MW-520 is located on COM4 and operates at a baud rate of 4800.

 **Warning:** If the GPS device gets unplugged from the serial port or the serial port gets closed, it is **strongly recommended** that TxMessenger be restarted to ensure that data is being sent and received properly. Failure to restart TxMessenger may result in the GPS not responding and/or TxMessenger running slowly due to its attempts to query a missing serial port.

 **Warning:** If the serial port is removed (such as removing a PCMCIA serial port adapter) while TxMessenger is running, the operating system and/or TxMessenger may become unstable and stop responding. It is **strongly recommended** that TxMessenger be restarted to ensure that data is being sent and received properly. Failure to restart TxMessenger may result in TxMessenger not responding.

22.4 Communicating with the GPS Device

After the GPS device is connected to the computer, check that TxMessenger communicates with the device. Here is a script that queries the GPS device for its version number and displays the result on the desk:

```
NEW MESSAGE;
SHOW "DESK";
GPS WRITE ">QVR<";
GPS READ GPSVersion "<" 750;
IF GPSVersion EMPTY;
     GPSVersion="GPS device not responding.";
ENDIF;
SET FIELD 1 TO GPSVersion;
DELETE GPSVersion
```

If nothing is displayed, make sure that the GPS device is powered on and is properly connected to the serial port. Make sure that the correct serial port has been specified by the `GPSPort` setting. Then check the `GPSBaud` and other GPS settings. Try setting the baud rate to a low value and then increase it until the GPS device responds to a command.

22.5 Sending GPS Data to the Host

To send GPS data to the host, the data is usually stored in the `GPSData` setting. For example, the script

```
GPS WRITE ">QPV<";
GPS READ GPSData "<" 750;
TRANSMIT GPS
```

will first send the QPV command to the GPS device, have up to 750 milliseconds to read data from the GPS device up to and including the '<' character, and then transmit the data if the '<' character was received from the GPS device.

 **Important:** The TX-protocol specifies that non-compounded GPS data may be sent on registers 0x46 or 0x48. The setting `GPSTransmitRegister=0x46` by default. If the host does not receive non-compounded GPS data, try changing this setting value to 0x48.

22.6 Sending Compounded Messages to the Host

GPS data can be transmitted with status, coded, emergency, or text/form data.

22.6.1 Sending Status Messages with GPS Data

To send GPS data with a status message, a status button's script could be:

```
ButtonStatus1Script= GPS WRITE ">QLM<";
                     GPS READ GPSData "<" 750;
                     TRANSMIT STATUS 1
```

The above script sends a status 1 message along with GPS data.

22.6.2 Sending Coded Messages with GPS Data

To send GPS data with a coded message, a coded message button's script could be:

```
ButtonCoded1Script= GPS WRITE ">QPV<";
                    GPS READ GPSData "<" 750;
                    TRANSMIT CODED 4
```

The above script sends a coded 4 message along with GPS data.

22.6.3 Sending Emergency Messages with GPS Data

To send GPS data along with the emergency message when the emergency button on the Motorola MW-520 display panel is pressed, this script could be used:

```
KeyMW5200Script=    GPS WRITE ">QID<";
                    GPS READ GPSData "<" 750;
                    TRANSMIT EMERGENCY
```

22.6.4 Sending Text/Form Data with GPS Data

To send GPS data with text:

```
ButtonOnscreen1Script= GPS WRITE ">QDL<";
                       GPS READ GPSData "<" 750;
                       TRANSMIT
```

This sends GPS data along with the contents of the desktop when the first onscreen button is pressed.

To send GPS data with a particular form, a button can have the script:

```
ButtonOnscreen2Script= IF DeskFormName="TA";
                         GPS WRITE ">QPV<";
                         GPS READ GPSData "<" 750;
                       ENDIF;
                       TRANSMIT
```

The above script sends GPS data along with the contents of form TA when the second onscreen button is pressed.

22.7 GPSReadScript

By default:

```
GPSReadScript=
     TRANSMIT GPS;
     GPS READ GPSData "<" 750;
     IF GPSData EMPTY;
        GPS READ GPSDiscardedData "." 750 FALSE;
        IF GPSDiscardedData = ".";
           GPS READ GPSDiscardedData "." 750;
        ENDIF;
        DELETE GPSDiscardedData;
     ELSE;
        TRANSMIT GPS;
     ENDIF;
```

This setting is executed whenever there is any data available from the GPS device that wasn't removed by a prior script.

First, `TRANSMIT GPS` transmits any GPS data that is already stored in the `GPSData` setting. If there isn't any available, it will do nothing.

Next, `GPS READ GPSData "<" 750` reads data available from the GPS device up to and including the "<" character and has up to 750 milliseconds to do so.

If the "<" character was found and stored in `GPSData`, then the data is transmitted. Note the `GPSData` setting is automatically cleared upon transmitting, so this script flushes cadence messages from the GPS to the modem on a regular basis.

If the "<" character is not found, `GPS READ GPSDiscardedData "." 750 FALSE` reads data available from the GPS device up to and including the "." character and has up to 750 milliseconds to do so, but does not clear the data from memory. `IF GPSDiscardedData = "."` checks to see if only a "." was read in. If a "." is the only data read, then `GPS READ GPSDiscardedData "." 750` clears the data from memory. In either case, `DELETE GPSDiscardedData` deletes the temporary setting. This section of the script may be necessary when the GPS device returns a "." when it is unable to receive data from GPS satellites.

 **Warning:** Altering the GPSReadScript in such a way that it fails to regularly process GPS data may cause the buffer to overflow. Under those circumstances, some GPS data may be lost and partial GPS results may be received and transmitted.

22.8 GPSTransmitScript

By default:

```
GPSTransmitScript=  GPS WRITE ">QPV<";
                    GPS READ GPSData "<" 750;
```

This is the script used in the previous examples to get data from the GPS device. Using this setting along with the `RUN` script command can simplify button scripts. For example, the script:

```
ButtonOnscreen1Script= IF DeskFormName="TA";
                          RUN GPSTransmitScript;
                       ENDIF;
                       TRANSMIT
```

will do the same thing as the previous example, but is shorter. This can be especially useful if a long script is required to get the GPS data needed.

23 Tray Panel

23.1 General

Beginning with version 2.5, TxMessenger has the capability of displaying a Tray panel. The tray panel is useful for creating and displaying custom buttons and bitmaps.

23.2 Displaying the Tray Panel

The value of the setting `ButtonTrayPanelShown` determines when the tray panel is displayed. If `ButtonTrayPanelShown=Always`, then the tray panel is always displayed. If `ButtonTrayPanelShown=Desk`, then the tray panel is only displayed when the Desk is displayed. If `ButtonTrayPanelShown=Never`, then the panel is never displayed.

Using the `SHOW "TRAY"` command will show the tray using the entire window, hiding all other panels. Press the ESC key to clear the tray from the window.

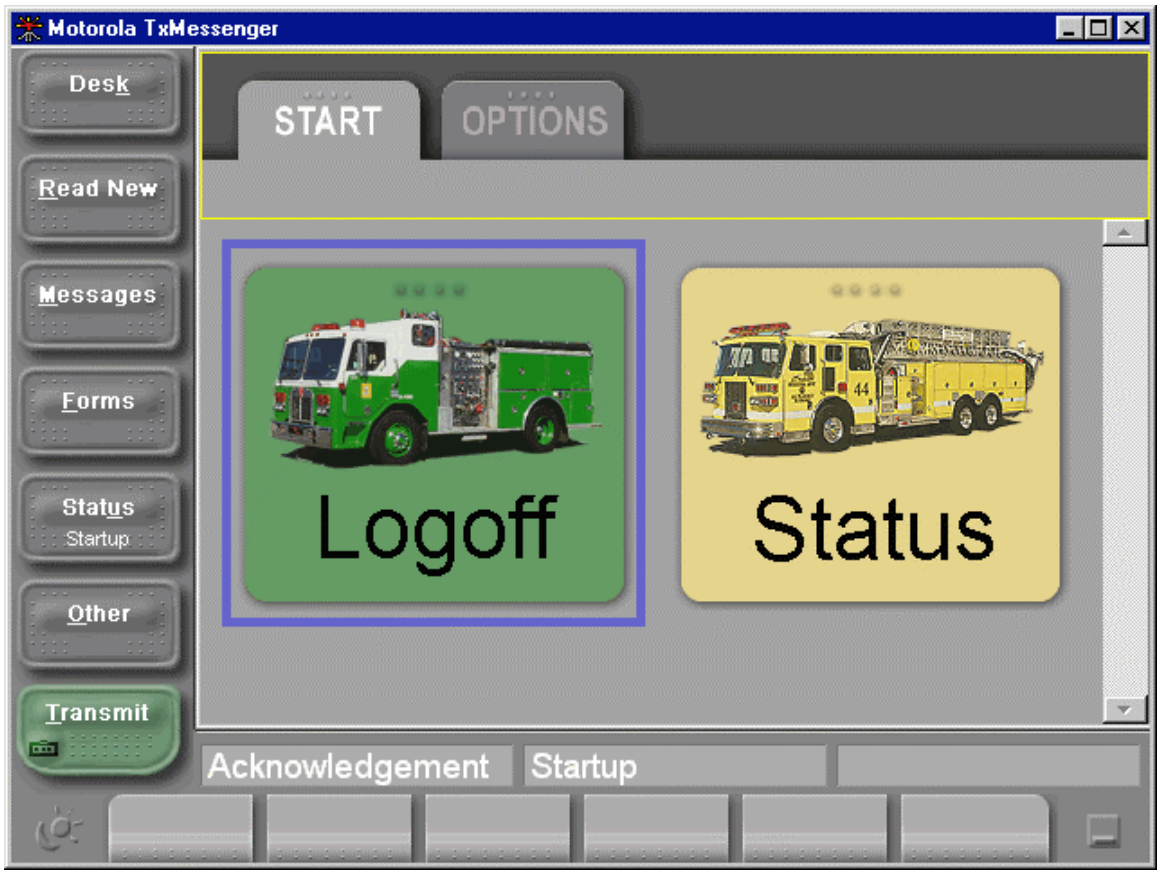
23.3 Location

The tray can be displayed at the top or bottom of the desk area. By default:

```
ButtonTrayLocation=Bottom
```

In the picture below, the tray panel is located at the bottom of the desk area. The tray panel is the area highlighted by a yellow rectangle.

If `ButtonTrayLocation=Top`, then the tray panel is placed at the top of the desk area. The tray panel is the area highlighted by a yellow rectangle.



23.5 Tray Panel Buttons

Tray panel buttons can be defined by using the `ButtonTray#Position` to define a "clickable rectangle", or button. The setting `ButtonTrayPanelPosition` is used to define the button's position, width and height. In the picture below, `ButtonTrayPanelBorder=BLUE`, so all of the defined buttons have a blue border around them.

Note: If the button number denoted by the # symbol does not fall between the values specified by `ButtonTrayPanelFirstButton` and `ButtonTrayPanelLastButton`, then the button will not be displayed on the tray panel.

In this particular case, a bitmap has been displayed in the tray panel by setting `ButtonTrayPanelPicture` to a picture of a keyboard. Then, buttons were defined, allowing a script to be executed when a button is activated.

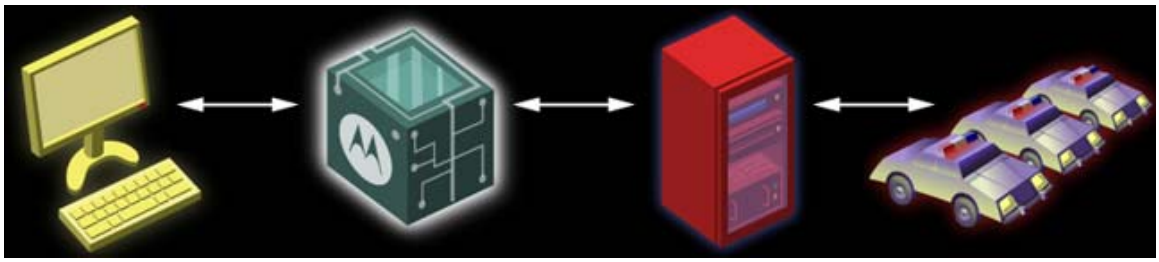
23.6 Tray Panel Scripts

When a tray panel button is activated, TxMessenger will execute the script defined by `ButtonTray#Script`, where # is the setting number for the button activated. If no script is defined, TxMessenger will then run the script defined by `ButtonTrayPanelScript.`

Appendix A: TxEncryptor

A.1 General

Beginning with version 3.5, TxMessenger has the capability of encrypting and compressing messages to and from the host. This is done with a separate proxy server inserted between the Host and the RNC. Below is a rough depiction of the inbound/outbound flow of messages.



```
Host ----- TxEncryptor ----- RNC ----- Clients
```

Warning: The TX protocol does not have application-level ACKs, only network level ACKs, and therefore does not have the elements necessary to guarantee end-to-end message delivery. TxEncryptor must remain compatible with the TX protocol in order to be compatible with existing TX clients and TX hosts; therefore TxEncryptor cannot guarantee message delivery.

A.1.1 General Features

- TxEncryptor is transparent and requires no changes to the Host (other than having the TX Host point to TxEncryptor's IP address rather than the RNC's IP address).
- TxEncryptor is compatible with existing mixed fleets of different TX applications and other applications, which allows for clients to be upgraded one at a time as desired.
- Messages from older TX-protocol clients will seamlessly pass through (unaltered).
- TxEncryptor automatically encrypts and compresses VLM messages to new TxMessenger clients as sent from the TX Host (outbound).
- TxEncryptor automatically decrypts and decompresses VLM messages from new TxMessenger clients before they get to the TX Host (inbound).

A.1.2 Compression

The TX protocol has very poor run-length encoding compression. TxEncryptor introduces new compression capable of dramatically reducing message/form delivery time and significantly increasing available bandwidth since TX VLM messages are ASCII text based and often times in all upper case letters. Lab results have seen VLM messages reduced to $^{1}/_{20}$th of their original size.

In practice, even a 50% overall reduction provides substantial benefits. For large test messages, lab results have shown up to 5 times improvement in throughput speed and 10 times reduction in bandwidth usage.

A.2 Minimum System Requirements

- **Processor:** Dual 1.8 GHz
- **OS:** Windows 2000 Server
- **Hard drive size:** 18 GB
- **Hard drive type:** Dual Channel Ultra 320 SCSI
- **Memory:** 1 GB
- **Power Supply:** 2 (one redundant, hot-swappable)
- **NIC:** 10/100 Mb Ethernet card
- **Video Card:** 8 MB

A.2.1 RNC Requirements

TxEncryptor requires a single RNC 3000 connected via TCP/IP.

- NCP or RNC 6000 may require a system upgrade
- Not for hosts connected via serial port
- No built-in support for redundant or multiple RNCs
  - Alternate configurations can be considered on a customer-by-customer basis
- TxEncryptor not for WNGs
  - Use MWCS II for Windows 2000 instead

A.3 Installing TxEncryptor

A.3.1 Disk and Operating System Security Rights

The installer creates directories, copies files, adds fonts and a startup file to the system, and modifies the registry. In any operating system with security or virus protection, it may be necessary to disable security/virus protection programs or logon with administrative rights to install TxEncryptor successfully.

A.3.2 Running the Installer

On the destination computer, run Setup.exe from the installation media to begin the installation procedure.

A.3.2.1 For Floppy Disks

Insert the first disk into the destination computer and run Setup.exe. The installer will ask for the remaining disks as needed.

The location of the installer program on most floppy drives would be A:\Setup.exe.

A.3.2.2 For Other Media

Insert the installation media into the destination computer and run Setup.exe from the \Motorola TxEncryptor Installer\Disk1 folder. Because the remaining disks are in the same folder as Disk1, the installer finds them automatically as needed without asking.

If the installer displays an error message, crashes, or continually asks to swap between disks, it is likely that the installer was either not copied over correctly or that the media is bad. Try reformatting the media and recopying the installer.

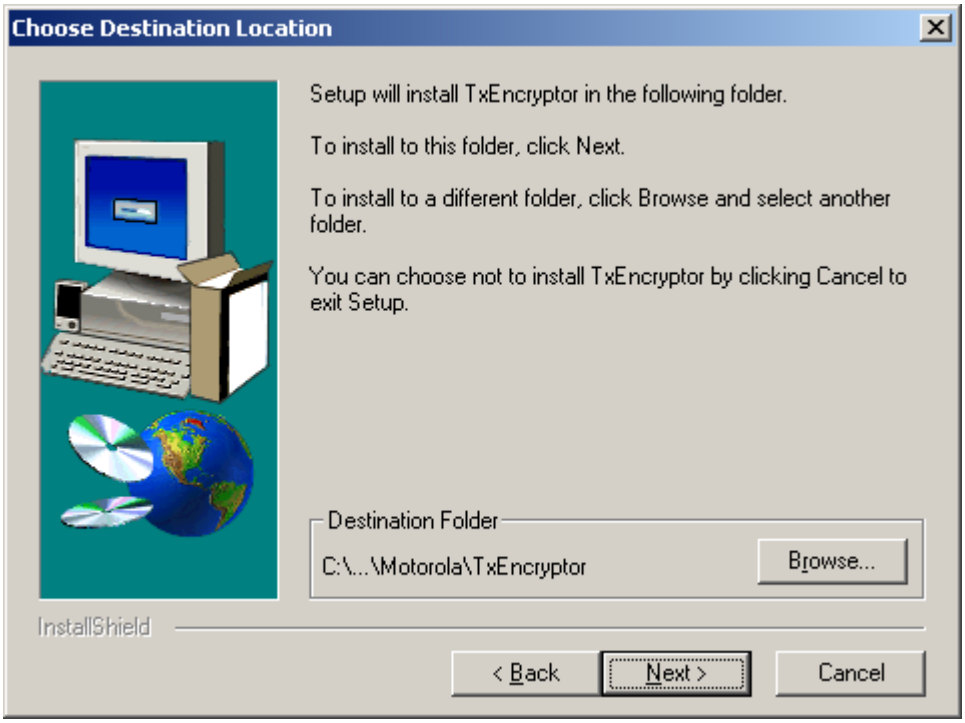
If the TxEncryptor logo in the installer background appears washed out or unattractive, the computer's display is probably less than the required thousands of colors. Follow the instructions on changing the display colors as described in section 2.6 of this document.

A.3.3 Release Notes

The release notes contain more recent information than this document. Before continuing, read the release notes. The release notes are also available on Disk 1 as `Release Notes.txt`.

A.3.4 Destination Directory

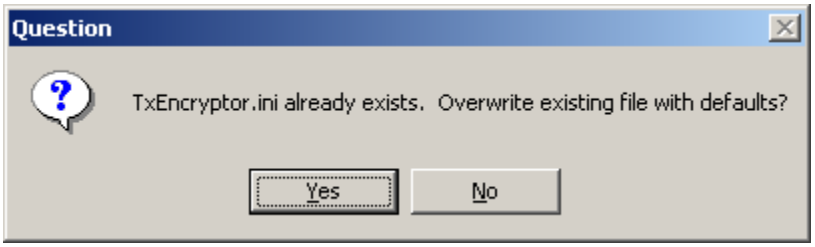
`C:\Program Files\Motorola\` is the default installation path provided by the TxEncryptor installer and is referred to throughout this section. Note that a different destination location can be chosen during installation of the TxEncryptor software (or other software packages), in which case the reader should substitute their specific destination paths accordingly in all examples.



The destination directory shouldn't be changed unless necessary. It is easier to find, upgrade, and maintain TxEncryptor if it is located in the default directory.

A.3.5 Overwriting an Existing TxEncryptor Directory

If TxEncryptor.ini already exists in the destination directory, the installer asks whether the existing TxEncryptor.ini should be replaced.



If the TxEncryptor.ini file is out of date or has become corrupted and unreadable, then it can be replaced by clicking Yes. Otherwise, clicking No can retain the current settings.

The contents of the Pictures, Scripts, and Sounds folders on Disk 4 of the installer are always copied to the destination directory. The files on Disk 4 will replace the files of the identical names in the destination directory without asking, but files with different names in the destination directory will remain unchanged. Therefore before installation:

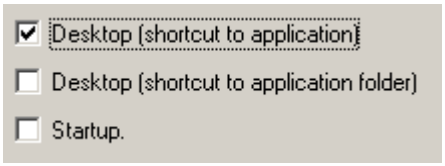
- Manually delete files in the destination directory that are no longer desired.

- Make a backup copy of files in the destination directory that should remain unmodified (i.e. customized scripts). After installation, manually copy those desired files back. This is only necessary for files in the destination directory that have the same name as files copied over by the installer. Files with different names are not overwritten or deleted by the installer.

Click next to display the list of currently selected settings for this installation. Click next again to begin installation of TxEncryptor.

A.3.6 Adding TxEncryptor to the Desktop and Startup Group

The TxEncryptor icon is always added to the Windows Start→Programs→Motorola Menu. In addition, near the end of the installation process, the installer provides the following additional choices:



- Adding the TxEncryptor program icon to the Windows Desktop makes running TxEncryptor more convenient for the user.

- Adding a TxEncryptor application folder shortcut to the desktop makes it more convenient for maintenance.

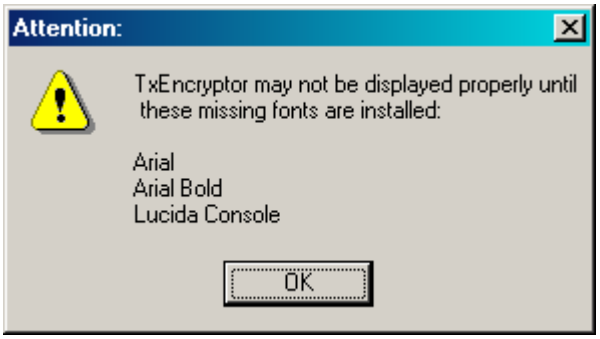
- Adding TxEncryptor to the Windows Startup Group makes TxEncryptor run automatically every time Windows starts up.

A.3.7 Restart the Computer after Each Install or Uninstall

Always restart the computer after any uninstall or install of software. If the computer is not restarted between uninstalling and installing, the installation may not be successful.

A.3.7.1 Font(s) Missing

If the computer hasn't been restarted, the required fonts may not be available. TxEncryptor displays a warning whenever a required font isn't available:



If restarting doesn't correct the problem, check the Windows Font folder (Start→Settings→Control Panel then open Fonts) to make sure that the following fonts are installed:

Font Name	Filename
Arial	`Arial.ttf`
Arial Bold	`Arialbd.ttf`
Lucida Console	`Lucon.ttf`

If the fonts are installed, open the font in the Windows Font folder by double clicking the font file to view a sample of it. This usually causes the operating system to recognize the font and make it available to applications.

Setting `FontRequiredMissingText` to blank in TxEncryptor.ini can disable the warning.

A.4 After TxEncryptor Installation

A.4.1 Startup Steps

- Add TxEncryptor to the same LAN as the RNC and the TX Host
- Configure the Host to point to TxEncryptor's IP address instead of the RNC's IP address. Note: If the RNC's IP address in the Host can't be changed, provide the RNC with a new IP address and use the RNC's original IP address for the TxEncryptor IP address.
- Open TxEncryptor.ini
  - Add the same four license settings found in the TxMessenger.ini
  - TxEncryptor cannot run without valid license settings
  - Add the setting `RNCIPAddress` and include the correct RNC IP address
- Startup RNC
- Startup TxEncryptor
- Startup Host
  - For sites with multiple hosts, it is strongly recommended that after starting both the RNC and TxEncryptor, to start up just one host first. After that host has connected, wait until the traffic has dropped and then repeat the process if there are other hosts.
- On each TxMessenger client you will need to add the following settings:
  - If only compression is desired:
    - `TxEncryptorEncryptionEnabled = FALSE`
    - `TxEncryptorConnectToPort1 = TRUE`
  - If both compression and encryption is desired:
    - `TxEncryptorConnectToPort1 = TRUE`

 **Warning:** Running TxMessenger/TxEncryptor without compression may result in very large messages being undeliverable due to the addition of the TxEncryptor header.

A.4.2 TxEncryptor Processor & Thread Values

The number of processors and threads is displayed on TxEncryptor's Main screen in the lower left corner. If the number of processors is incorrect, add the following settings to the .ini:

```
CompletionPortThreadPoolSize = number of processors
CompletionPortNumberOfConcurrentThreads = number of
processors * 2
```

A.4.3 TxEncryptor Exit Warning

To set up TxEncryptor to display a warning dialog box upon exit, add this setting to the ini:

```
Exit = Alert
```

The default exit warning is "Are you sure you want to exit?". If you wish to change this message, add this setting to the ini:

`ExitMessagePrompt = *Your Message* (with or without quotes)`

A.4.4 Customizing Built-in Scripts

If a script is placed in the scripts folder with the same name as a built-in script, TxEncryptor will automatically try to run the user-defined script. Conversely, if a user-defined script is missing, TxEncryptor will automatically look for and then run the built-in script with the same name.

The best example of how to use this to your advantage is to take a look at the scripts in the Goodies\Scripts\ TxEncryptor built-in scripts folder on your TxMessenger installation CD. These scripts are built-in and are routinely run while TxEncryptor is active.

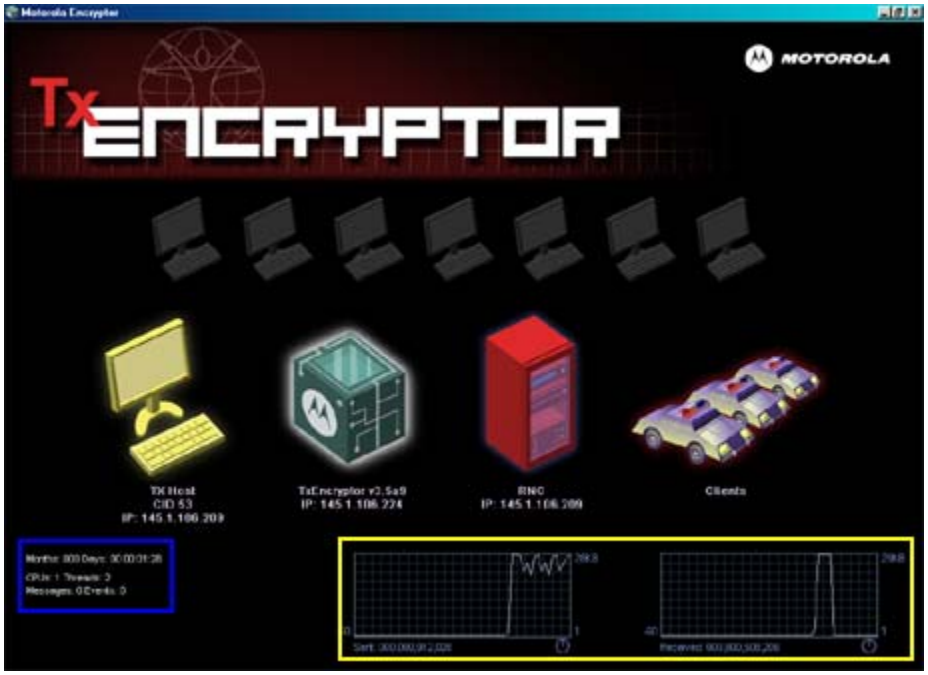
- `BoxInitializeScript.txt`
- `HostEventScript.txt`
- `InitializeScript.txt`
- `LIDEventScript.txt`
- `LoadLIDAliasesScript.txt`
- `RNCStatusChangedScript.txt`
- `StartupScript.txt`
- `TrayCleanupScript.txt`
- `TxEncryptorStatusChangedScript.txt`

Inside the `InitializeScript.txt` file are several `RUN FILE` *`filename`* commands that reference the other scripts listed above. These script files can be customized and placed in the TxEncryptor\Scripts folder. Be sure that the folder you place your customized script is the same folder described in the `RUN FILE` *`filename`* path in the `InitializeScript.txt`. The next time TxEncryptor calls one of these scripts; the customized script will be run instead of the built-in script.

A.5 Using TxEncryptor

A.5.1 TxEncryptor Main Screen

TxEncryptor runs on a separate proxy server with its own settings file (INI). Below is TxEncryptor's main screen.



The yellow icons represent each Host with their respective CID below. A maximum of eight hosts can be connected at one time. For best performance and compatibility, only TX hosts should point to TxEncryptor's IP address. Other Hosts should still point directly to the RNC.

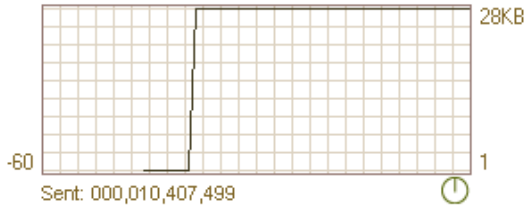
Also displayed below the first connected host is the RNC's IP address. The RNC is represented by a red icon with its IP address below. The trio of police cars represents communicating clients. TxEncryptor is the box in between the host and the RNC. Its version number along with its IP address (if provided by the setting `TxEncryptorIPAddress`) is displayed below.

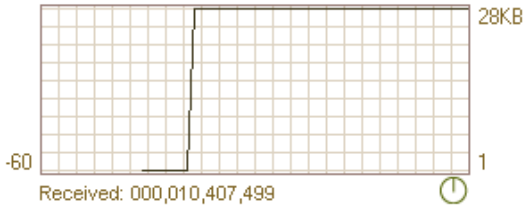
The area highlighted in blue above displays TxEncryptor's connection duration, the number of CPUs detected in the TxEncryptor box and the number of Threads created. The number of Threads is always twice the number of CPUs. The Messages field keeps track of the number of queued messages and the Events field keeps track of the number of queued update events.

A.5.2 Sent/Received Graphs

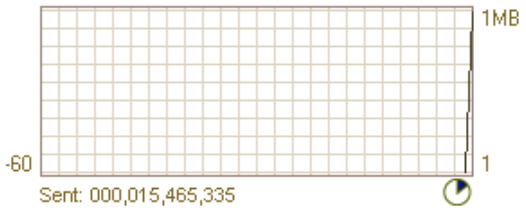
The area highlighted in yellow above contains two graphs showing the number of bytes sent and received. The viewing mode can be toggled by clicking anywhere inside either graph. There are three different viewing modes; minute, hour, or day. Each graph has a peak or maximum value for number of bytes the graph can plot. This value is configurable for each graph. These graphs are only to be used as a general idea as to the amount of bytes sent and received. If you are looking for an accurate count, a third-party application should be used.

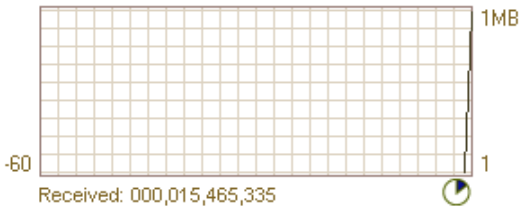
Minute Mode = kilobytes sent and received per minute



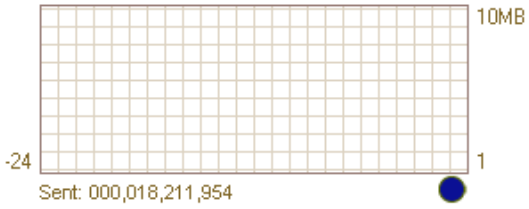


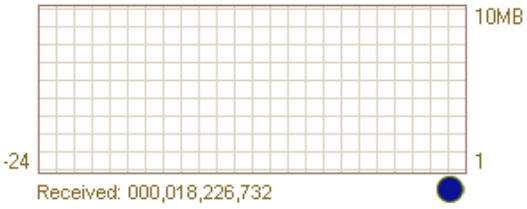
Hour Mode = megabytes sent and received per hour





Day Mode = megabytes sent and received per day





A.5.3 Connections / Disconnections

Whenever TxEncryptor detects a host or RNC disconnection, the corresponding icon will disappear. Once a host is connected and clients are detected, the client icon will appear. However, if the host disconnects or its connection is lost, the client icon will remain. By default, the RNC is pinged every five seconds and if TxEncryptor doesn't receive a response from the RNC in one second, the RNC icon will disappear. During normal operation, the RNC icon may flash. This is due to the ping timing out before receiving a response. Both the ping interval and the duration to wait before timing out are configurable, but the less time between pings the quicker a connection failure can be detected.

TxEncryptor is passive when it comes to connections and disconnections. It will not initiate connections or disconnections. Only after the Host initiates a connection with TxEncryptor will TxEncryptor initiate a connection to the RNC. The same holds for disconnections. TxEncryptor will only initiate a disconnection after the RNC or the Host initiates a disconnection first. The main exception is when the user elects to quit TxEncryptor. When this occurs, TxEncryptor will initiate disconnections from both the Host and the RNC.

A.5.4 Removing TxEncryptor from the System

In the event that TxEncryptor needs to be removed from the system, configure the Host to connect directly to the RNC IP address.

If the TxMessenger clients still have enhanced services enabled (encryption, compression, etc, see A.4.1) then non-TX protocol messages (such as connection requests with TxEncryptor headers) will be received by the Host when those clients try to connect to TxEncryptor. Although the Host should simply ignore these messages (since they don't start with 'TX'), some Hosts may be poorly written and may not handle the messages properly. In this case, TxEncryptor service settings `TxEncryptorConnectToPort1` and `TxEncryptorSessionHeaderSent` should be disabled in the clients.

 **Warning:** Failure to disable TxEncryptor services once TxEncryptor is unavailable may cause incompatibility, crashes, or undefined behavior.

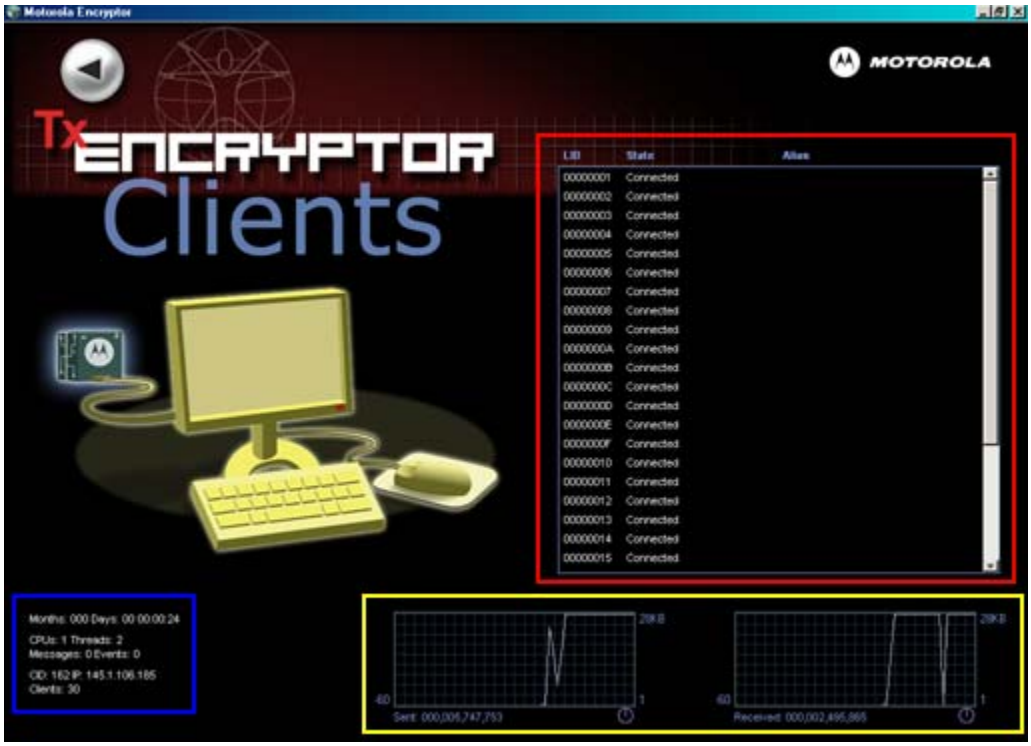
 **Warning:** If TxEncryptor is removed from the system, or if the number of users exceeds the number of licenses, and TxMessenger starts up with encryption enabled, TxMessenger will eventually give up and start sending clear text messages. However, if TxEncryptor is removed from a system while TxMessenger is already running and connected, then TxMessenger must be exited and rerun to operate with clear text.

A.5.5 Orphaned Sessions

TxEncryptor associates each session with a modem ID. If a user disconnects their modem, but is unable to cleanly disconnect from TxEncryptor, for instance, if a user has roamed out of range and exits TxMessenger, the active session will still be open for that modem ID. If a second user then connects to this modem, TxEncryptor will grant that user an active session since it will be able to associate the previously orphaned session with this modem ID. Properly exiting TxMessenger can help reduce the amount of orphaned sessions.

 **Warning:** It is recommended that customers purchase a number of TxMessenger/TxEncryptor licenses equal to the number of modems they own. Otherwise, the modem IDs of users that have not properly disconnected will continue to monopolize that license rather than sharing it with a new modem that is trying to connect.

A.6 TxEncryptor - Clients Screen



A.6.1 General

Clicking any of the Host icons on the Main screen brings up the Clients screen for that specific Host. As clients communicate, the modem ID is automatically added to the client list. There is no need to manually enter modem IDs. The client list, highlighted in red, contains the modem's LID (logical identifier), State and Alias. To return to the Main screen, simply click the back arrow in the top left corner.

A.6.2 Client List

The State column shows the current state of each client as Connected, Disconnected, or Pass Through. Connected means that the client has successfully completed the handshake with TxEncryptor and TxEncryptor's services are now available to that client through the connected session. Disconnected means the client has successfully closed its session with TxEncryptor. Pass Through means that messages to and from the client are not encrypted or compressed (for example, older TX clients). Encryption and compression automatically engage or disengage based on the client. Also, each modem ID can be associated with a label (through the use of the setting `ClientsLID#Alias`), like 'Car 17', if desired. That label is listed in the Alias column.

A.6.3 Sent/Received Graphs

Highlighted at the bottom in yellow, are the same two graphs from the Main screen displaying the total bytes sent and received.

A.6.4 Host Information

The bottom left corner, highlighted in blue above, contains specific information for the selected Host.

- Host's CID
- Host's IP address
- Number of Clients connected to the Host

The other information in this corner is the same that is found on TxEncryptor's Main screen.