



Inova OnAlert™ Display

Scripting Guide

CONFIDENTIAL



Inova OnAlert Display
Scripting Guide

Revision 003

CONFIDENTIAL

While reasonable efforts have been taken in the preparation of this document to ensure its accuracy, Inova Solutions, Inc. assumes no liability resulting from any errors or omissions in this manual, or from the use of the information contained herein.

© 2011 Inova Solutions, Inc.
110 Avon Street
Charlottesville, VA 22902
434.817.8000
www.inovasolutions.com

Table of Contents

1. Introduction	1
1.1. Messaging Capability	1
1.2. Other Documentation.....	1
2. Overview of Message File Organization	3
3. Create a Simple Message for the Display	4
3.1. Configure the Display	4
3.2. Connect to the Display	4
3.3. Create a Test Message.....	4
3.3.1. Use Interactive Mode.....	4
3.3.2. Use Scripted Mode.....	5
4. Command Exploration	6
4.1. The localmsgs Command	6
4.2. The libmsgs Command.....	6
4.3. The startupmsgs Command	6
4.4. Next Steps	7
5. Best Practices	8
5.1. Security	8
5.2. Protecting Flash Memory.....	8
5.3. Direct Folder Access	9
6. Message Factory	10
6.1. Overall Process	10
6.2. Create a Message using Message Factory.....	11
6.2.1. Sample Message	11
6.2.2. Start a New Message Recipe	11
6.2.3. Create a Message Region.....	12
6.2.4. Add Text Contents.....	12
6.2.5. Compose Additional Add Region and Add Text Commands	13
6.2.6. Generate the Message	14
6.2.7. Region Specifications.....	14
6.2.8. Customize a Message for Larger Display.....	15

CONFIDENTIAL

6.3. Command Line Flags.....	16
6.3.1. About the "--".....	17
6.3.2. Error Handling Notes.....	17
6.4. Positional Parameters for Individual Commands.....	17
6.4.1. Parameters for Starting a New Recipe (<i>new_msg</i>).....	17
6.4.2. Parameters for Adding Message Regions (<i>add_region</i>).....	18
6.4.3. Parameters for Adding Text (<i>add_text</i>).....	19
6.4.4. Parameters for Adding Data Fields (<i>add_df</i>).....	19
6.5. Two-line Example Message with Time Date Field.....	22
6.6. Message Status.....	22
Appendix A - More on the localmsgs Command.....	24

Figures

Figure 1: Sample Message Factory Commands.....	11
Figure 2: Sample Display Message.....	11
Figure 3: Sample message for larger display.....	15
Figure 4: Using the --.....	17
Figure 5: Two Line Sample Message.....	22
Figure 6: Message Status Report.....	22

Tables

Table 1: Command Line Flags.....	16
Table 2: <i>new_msg</i> Parameters.....	18
Table 3: <i>add_region</i> Parameters.....	19
Table 4: <i>add_text</i> Parameters.....	19
Table 5: <i>add_df</i> Parameters.....	21

CONFIDENTIAL

1. Introduction

The Inova OnAlert™ Display can be controlled either by using the provided Inova OnAlert Server software or from a command script generated by partners or customers as described in this document. Both of these options use the same command oriented scripting interface, usually via an encrypted Secure Shell (SSH) connection; however, these options should be used separately. For example, scripting messages while Inova OnAlert Server software is connected to the same display(s) may have unintended consequences. The Inova OnAlert Display is based on an embedded Linux operating system, so the command interface will be particularly familiar to readers experienced with Linux or Unix commands.

1.1. Messaging Capability

The Inova OnAlert Display is capable of displaying messages in a variety of presentation styles, fonts, and colors.

The simplest message type uses the entire display as one text region; it typically uses the scroll up presentation mode and a font selected so that two lines of text are visible on the display. A single sentence or a long paragraph can be transmitted to the display as one text message. The display automatically performs a word wrap function on the text and the text moves in a movie credit upward scroll on the display and repeats until the message is deleted.

More complex messages can be scripted that use two text regions (i.e., text on the top line might be static on the display while text on the bottom line is scrolling either horizontally or vertically).

Further, it is possible to create messages that use system variables (i.e., date or time) to embed these elements into a message. The display also has a default time and date message that can be enabled by enabling Simple Network Time Protocol (SNTP), either with the “config edit” command or through the display’s internal website. If this `_default_time.llm` message is inadvertently later deleted, it can be restored by disabling and then re-enabling SNTP.

1.2. Other Documentation

For additional information not found in this Scripting Guide, refer to the *Inova OnAlert Display Installation and User Guide* and the *OnAlert Server Software Installation Guide*, which cover:

- Hardware installation and network configuration
- Inova OnAlert Server software installation and configuration
- Accessing the display through its internal website

CONFIDENTIAL

- Telnet and SSH connections

CONFIDENTIAL

All claims based on information publicly available at time of printing. All other product or service names mentioned in this document may be trademarks of the companies with which they are associated.

© 2010 Inova Solutions. | All rights reserved. 6.17.2011 | page 2

2. Overview of Message File Organization

The messaging capability of the display is best understood in terms of a file folder model with three folders, as described below:

- **Local Folder** - Any message files (i.e., those with file extension .llm) located in this folder will be stored to play on the display. The folder is scanned for new messages to play at least twice per second. This folder is completely cleared by a power cycle.
- **Library Folder** - The contents of the Library folder persist through a power cycle. Message files may be created or placed in the Library Folder for storage and then be copied as needed into the Local Folder for display with a simple user script run from the command console.
- **Startup Folder** - Message files that are located in this folder are automatically copied to the Local folder at display boot time. This offers a way to pre-configure a message or set of messages to always play after a display power cycle. The files in the Startup folder are not cleared by a power cycle.

CONFIDENTIAL

All claims based on information publicly available at time of printing. All other product or service names mentioned in this document may be trademarks of the companies with which they are associated.

© 2010 Inova Solutions. | All rights reserved. 6.17.2011 | page 3

3. Create a Simple Message for the Display

3.1. Configure the Display

Follow the instructions in the *Inova OnAlert Installation and User Guide* for getting the display established on your network. The display supports DHCP and static IP addresses. The simplest way to configure the network properties of the display is via the onboard web site as described in the manual.

Note that if you configure the displays with static IP Addresses, your displays cannot connect to a time server with a DNS name. The displays must connect with a static IP address.

3.2. Connect to the Display

Use an SSH client program to connect to the display using the admin user ID and the published password. There is a section in the *Inova OnAlert Installation and User Guide* about how to access the display command console, along with a complete list of commands. *Note: The commands are also available by typing 'help' at the command prompt.*

The admin login offers user level access to the Linux command console. The best way to gain familiarity with the scripting capability of the display is to start by manually using the scripting commands to create messages and then exploring scripted or programmatic operation.

3.3. Create a Test Message

3.3.1. Use Interactive Mode

Note that this portion of this document is written for an audience with some familiarity with command line scripting.

The steps below will walk you through building a message on the display and placing the message file into the Local Folder, where it will immediately begin playing on the display.

1. From the command line, type the compose command without any parameters:

```
localmsgs compose
```
2. The display will respond by prompting you for all of the necessary fields used to create a message.
 - a. **Filename** – enter a filename (e.g., test_one). Specification of the file extension is not required in this interactive mode; the required .llm extension is automatically added.

CONFIDENTIAL

Note that referring to the message file in subsequent commands will require explicitly adding the .llm extension.

- b. **Text** – enter a line of simple message text.
 - c. **Font Size** – select a font size. For an X2 or M2 display, the font size 8 is roughly half the display height; 16 is the full height.
 - d. **Bold** – the default selection is normal text.
 - e. **Presentation Mode** – unless you are sure that your text will all fit on the display through experimentation, select the Scroll Up mode.
 - f. **Dwell Time** – zero is suggested. A non-zero entry will hold the message (or the part of it that fits) on the display for some length of time at the middle of the presentation cycle; this is best used only if you are sure that all of the text will fit on the display at the same time.
 - g. **Color** – select a color.
 - h. **Sound** – a sound can be generated in displays that are equipped with speakers.
3. The command exits with a note that the message has been created. The message will play on the display in a round robin fashion with any other local messages.

You can verify that the message is in the folder by using the list command as shown below.

```
localmsgs list
```

You can delete the message by using the delete command as shown below.

```
localmsgs delete -f test_one.llm
```

3.3.2. Use Scripted Mode

All of the required information about a message can be put on a single command line, which makes the localmsgs command much more useful to a script developer.

```
localmsgs compose -f test_two.llm -c red -p scroll_up  
-s 8 -d 0 -t 'This is the second test message.'
```

You can verify that the message is in the folder by using the list command:

```
localmsgs list
```

You can delete the message by using the delete command:

```
localmsgs delete -f test_two.llm
```

CONFIDENTIAL

4. Command Exploration

The commands described in this section allow a script developer to create simple messages on the display to play now, later, or upon a power cycle. The script developer can also store messages into and delete messages from a persistent message library. A script based system using these commands will have complete control of basic display functionality.

4.1. The `localmsgs` Command

As described above, the `localmsgs` command is used to compose messages that will begin to play immediately. It is important to understand that this is because the message file generated is built in the Local Folder (i.e., the folder from which messages are played). Messages in the Local Folder are cleared by a power cycle.

The `localmsgs` command can also be used to return a list of the message files in the Local Folder, to copy messages from the library folder, or to display the properties of a message file. A complete description of the `localmsgs` command is available in Appendix A or by typing `localmsgs -h` at the command prompt.

4.2. The `libmsgs` Command

The `libmsgs` command is essentially identical in capability to the `localmsgs` command except that it acts on the Library Folder rather than the Local Folder. If you compose a message using the `libmsgs` command, then that message will be stored in the Library Folder rather than the Local Folder. A message in the Library Folder will not play on the display, but it will persist in the flash memory through a power cycle.

4.3. The `startupmsgs` Command

The `startupmsgs` command is also essentially identical in capability to the `localmsgs` command except that it acts on the StartUp Folder rather than the Local Folder. If you compose a message using the `startupmsgs` command, then that message will be stored in the StartUp Folder rather than the Local Folder.

A message in the StartUp Folder will not play on the display when it is created, but a message in that folder will persist in the flash memory through a power cycle. Moreover, when the display is power cycled, any message files in the StartUp Folder are automatically copied over to the Local Folder – at which time they will play on the display.

Note that the display boot process requires about 45 seconds.

CONFIDENTIAL

4.4. Next Steps

A basic script system can be built using the `localmsgs`, `libmsgs`, and `startupmsgs` commands as described above. A more advanced script system uses the contents of these two sections:

- *Section 5. Best Practices* - for best practices and important security information.
- *Section 6. Message Factory* - for a way to create message files that are more complex than those allowed by the `compose` command previously described.

CONFIDENTIAL

All claims based on information publicly available at time of printing. All other product or service names mentioned in this document may be trademarks of the companies with which they are associated.

© 2010 Inova Solutions. | All rights reserved. 6.17.2011 | page 7

5. Best Practices

5.1. Security

Inova Solutions recommends that script developers use SSH (Secure Shell) with a password of their own choosing for all communications with the display. Further, once the networking parameters of the display are set, the website and Telnet servers in the display should be disabled. This is important because the passwords for both the website and Telnet access are transmitted to the display unencrypted. All SSH traffic is encrypted.

Inova OnAlert Server software uses SSH and performs an automatic lockdown of the display, both changing the password and disabling all non-secure protocols.

5.2. Protecting Flash Memory

The persistent storage in the display is implemented using flash memory. This type of memory can be damaged if it is written too many times. The useful lifetime of flash memory is typically 100,000 cycles, which is large enough that the flash memory should not fail in normal operation. Unfortunately, it is possible to build a script that generates many writes to flash memory at computer assisted speed. Follow these guidelines to avoid unnecessarily degrading the flash memory on the display:

- When sending a message, use the `localmsgs` command to place it in the Local Folder, which is in the dynamic RAM memory and offers unlimited write cycles. This folder is not persistent through a power cycle.
- Do not unnecessarily write to the persistent folders (Library and Startup Messages). In particular, avoid a scripting mistake that writes to the same file repeatedly.
- While in script development, review on the number of flash writes being performed by reviewing the flash write counter in the file `/etc/config/flash.counter`.

In the example below, the `cat` command is used to review the number of flash writes. The time and date of the last time that the flash memory was written is also returned.

```
$ cat /etc/config/flash.counter
last_write=Mon Oct 5 15:16:21 UTC 2009
total_writes=12
```

CONFIDENTIAL

5.3. Direct Folder Access

The folder model described in Section 3 is intended to serve as an explanatory tool. When developing scripts, always use the `localmsgs`, `libmsgs` and `startupmsgs` commands to control message content.

Script developers should not try to access files in these folders or change the contents directly since unintended consequences may occur, particularly as there is no implicit guarantee that the internal implementation will be consistent from version to version.

CONFIDENTIAL

6. Message Factory

This section of the document is intended for people who are experienced with Linux scripting.

The Message Factory is used to construct messages for the Inova OnAlert Display; it allows more complex message options than the `localmsgs compose` command alone. For instance, a two line message can be created with static text of one color on the top line and scrolling text of a different color on the bottom line. Also, a message can be created that includes the time or date or both; this would dynamically update.

The Message Factory works by creating a temporary recipe file on the display in a folder named `tmp`. The `localmsgs` command is then used to create a message file from the recipe file, rather than from parameters on the `localmsgs` command line.

Note that the primary example in this section (Figure 1) is for the X2 series display, which is composed of an LED array 16 pixels high and 96 pixels wide. At the end are more examples, some of which are only applicable to larger displays (M4, M6, and M8 models).

6.1. Overall Process

The `message_factory` Application Programming Interface (API) is accessible from the console command interface and exposes most of the message formatting features of the display. The `message_factory` recipe file defines all the characteristics of a message for the display. It may take several `message_factory` commands acting on a single recipe file to obtain the desired message recipe. Then the message can be shown on the display using the `localmsgs compose` command specifying the recipe file. Once playing, its status can be viewed using the `localmsgs msgstatus` command (refer to Section 6.6).

There are four commands to use in building message recipes: `new_msg`, `add_region`, `add_text`, and `add_df`. The commands `new_msg`, `add_region`, and `add_text` are described in the following sections and are used in the primary example in Figure 1.

Note: The `add_df` command, for adding realtime data fields to a message, is somewhat more complicated but is also less frequently used; it is documented in the section "Parameters for Adding Data Fields" and demonstrated in the first additional example (see section 6.4 below).

CONFIDENTIAL

6.2. Create a Message using Message Factory

6.2.1. Sample Message

The following set of commands, run at a display command prompt, creates a two-line message that shows on the display immediately, assuming no messages of higher priority are already configured on the display. Each command is shown with the \$ prompt character to delineate each complete command in this example (i.e., enter each command as a single line entry even though some of them wrap to a second line in this example).

```

1 $ message_factory -f /tmp/mymsg_recipe -c new_msg -- panel
2 $ message_factory -f /tmp/mymsg_recipe -c add_region -n 1/2 -- 1 appear
  scroll_up fastest 20000 center bottom
3 $ message_factory -f /tmp/mymsg_recipe -c add_text -- 1 7 bold block normal
  yellow black none none 'BOMB THREAT'
4 $ message_factory -f /tmp/mymsg_recipe -c add_region -n 2/2 -- 1
  ribbon_left ribbon_left fastest 0 center bottom
5 $ message_factory -f /tmp/mymsg_recipe -c add_text -- 1 8 normal block
  normal red black none none 'An emergency has been reported in the building.
  Please proceed to the nearest exit and vacate the building.'
6 $ localmsgs compose -f inova_onalert_msg.llm -e /tmp/mymsg_recipe

```

Figure 1: Sample Message Factory Commands

The above example creates a recipe file in the location /tmp/mymsg_recipe and then the localmsgs compose command uses that recipe file to create the message that shows on the display. The commands in this example are detailed in the following sections. See Figure 2 for this sample message on a display that is 16 high by 96 wide:



Figure 2: Sample Display Message

6.2.2. Start a New Message Recipe

The first step required when using message_factory is to start a new message recipe. This is accomplished with a new_msg command (line 1 in Figure 1), such as:

CONFIDENTIAL


```
message_factory -f /tmp/mymsg_recipe -c new_msg --
critical
```

This creates a new recipe file called `/tmp/mymsg_recipe` using the optional `-f` flag. Leaving off the `-f` flag and its value would create a recipe file using the default location and name of `/tmp/llm_builder.recipe`.

This recipe uses the highest-available critical priority for the new message. The priority is optional, and will default to normal if not specified.

6.2.3. Create a Message Region

After the message recipe has been started, at least one message region must be created to specify the overall presentation properties for the content (e.g., entry mode, text speed, justification).

A message region defines an active area within the overall message dimensions that can contain text or data field items. Multiple message regions are allowed, and each one specifies presentation properties for the text content specified within its region specification.

Our example message has two regions (lines 2 and 4 in Figure 1). The first region is defined by the following command:

```
message_factory -f /tmp/mymsg_recipe -c add_region -n 1/2
-- 1 appear scroll_up fastest 20000 center bottom
```

The line number flag (`-n`) specifies which line out of the number of equally-sized lines this region will occupy. Using the `-n` flag with the `1/2` value indicates that this region is the first line of two equally sized lines. For our example display, this results in two lines, each 8 pixels high and 96 pixels wide.

For additional details about region specifications, refer to Section 6.2.5.

The presentation properties for the `add_region` command are specified as positional parameters; refer to Section 6.4 for more information.

6.2.4. Add Text Contents

Now that a message region has been defined, text contents can be added to that message region using the `add_text` command to specify text presentation properties such as font size, type, and color. Multiple `add_text` commands can be specified to allow for text with different font properties.

You can see this command in the line 3 of the sample message in Figure 1:

```
message_factory -f /tmp/mymsg_recipe -c add_text -- 1 7
bold block normal yellow black none none 'BOMB THREAT'
```

Since no region size is specified with this command, the region size is re-used from the last region size specification, given with the `add_region` command. A

CONFIDENTIAL

region size can be provided for *add_text* if needed, but it is often convenient to use the last size specified in the *add_region* or *new_msg* command.

The message text properties for font size, type, and color, are specified as positional parameters, as documented below in the section, *Parameters for Adding Text*.

The *add_text* (and *add_df*) commands will build message content sequentially in reading order in the recipe file.

6.2.5. Compose Additional Add Region and Add Text Commands

The *add_region* and *add_text* commands covered above only specified the first line of the message. The additional two commands specify the contents of the second line.

The second *add_region* command (line 4 in Figure 1) is:

```
message_factory -f /tmp/mymsg_recipe -c add_region -n 2/2
-- 1 ribbon_left ribbon_left fastest 0 center bottom
```

The line number flag (-n) is now used to specify this new region as the second of two lines equally sized lines within the message.

Different presentation properties are configured for this *add_region* command than were specified for the first one shown above. This creates a visible difference between the two regions to produce a visibly engaging presentation of the message contents.

Similarly, the second *add_text* command (line 5 in Figure 1) follows the second *add_region* command and provides the text contents and formatting of the second line of the example message:

```
message_factory -f /tmp/mymsg_recipe -c add_text -- 1 8
normal block normal red black none none 'An emergency has
been reported in the building. Please proceed to the
nearest exit and vacate the building.'
```

Like the first *add_text* command, this one re-uses the region size specified in the preceding *add_region* command; in this case, it is the second of two lines within the message.

The message text properties for font size, type, and color are specified as positional parameters. Refer to Section 6.4 for additional details.

Note that this message text would exceed the height or width of the region in which it is being displayed if it were statically displayed; however, the *ribbon_left* presentation mode determines its scrolling behavior and allows the entire text to display.

CONFIDENTIAL

6.2.6. Generate the Message

Once a recipe file has been created, it can be used to generate the message on the display. Given a recipe called `/tmp/mymsg_recipe`, the following command will turn it into a message called `mymsg.llm` on the display; the recipe file is deleted when this command is run:

```
localmsgs compose -e /tmp/mymsg_recipe -f mymsg.llm
```

After this command is issued, any errors in the recipe file will be reported. If there are no fatal errors in the recipe, the message `mymsg.llm` will be created and will appear on the Inova OnAlert Display.

6.2.7. Region Specifications

When creating regions, remember that:

- Region size specification can also be used with any of the following commands: *new_msg*, *add_region*, *add_text*, *add_df*.
- A message created with a given size specified in the *new_msg* command would only cover the specified region of the display. If no sizing is specified with the *new_msg* command, then default is to cover the entire face of the display.
- Any region size specification can be used for subsequent commands by default.

The **-r flag** specifies a region size using two coordinates (top and left positions) and two sizes (height and width) to specify a rectangular region size. For example:

- An **-r** flag with a value of `0,0,8,96` would define a region within the message size starting at the upper left of the message and spanning 8 high and 96 wide. This is another way to specify the top line the display.
- An **-r** flag with a value of `8,16,8,80` would define a region within the message size starting 8 below and 16 in from the upper left of the entire message, spanning 8 high and 80 wide.

The **-r** region size specification is more powerful than the **-n X/M** line abstraction, but it is also more complex and requires consideration of the actual size of the display and the message.

The **-n flag** in a *new_msg* or *add_region* command allows you to split the physical display face into a set of equally sized lines of text, each with a minimum height of 8 pixels.

CONFIDENTIAL

6.2.8. Customize a Message for Larger Display

For larger displays, a message can be split into smaller message regions with subsequent `add_region`, `add_text`, and `add_df` commands, using the `-r` and `-n` flags. (The message in Figure 1 can only be divided into two lines no matter which command is used because the minimum height of a line is always 8 pixels.)

Once the physical region of the message is defined (either using lines or regions), the line abstraction can further divide up the physical space when used with `add_region` or `add_text` commands.

These nested regions are sized relative to the overall physical space claimed for the entire message. The line flag is specified in terms of `-n X/M`, where `M` is the total number of lines and `X` is the specific line to use.

In the sample message in Figure 3, the top half of the 32-high display is used as one physical region for the entire message (`-n 1/2`). Since this display is 32-high, each of the two equally sized lines specified in the `new_msg` command is 16-high. Then, a relative region using just the bottom line of the two lines is added (`-n 2/2`). Note that in relation to the entire 32-high display, this would be the second line down from the top. The text is then also added to the second line.

```
message_factory -c new_msg -n 1/2
message_factory -c add_region -n 2/2 -- 1 ribbon_left ribbon_left medium 0
left middle
message_factory -c add_text -n 2/2 -- 1 8 bold profile normal yellow black
none none 'Inova Solutions'
```

Figure 3: Sample message for larger display

Be sure that you check the message; the `message_factory` command will catch most common sizing errors when using *lines* (`-n`), but will not provide any checking for generic region sizing (`-r`).

Note that the size of a line is independent of its presentation mode; a message specified with a scroll up behavior will scroll the entire text within the specified line size.

CONFIDENTIAL

6.3. Command Line Flags

The general form for command line flags is:

```
message_factory {flags} -- {positional parameters}
```

Refer to Table 1 for details about the command line flags.

Flag	Meaning	Explanation
-c	command	Can be: <i>new_msg</i> , <i>add_region</i> , <i>add_text</i> , or <i>add_df</i>
-f	recipe filename	File that is iteratively generated by the message factory. If this parameter is omitted, then a default of <i>/tmp/llm_builder.recipe</i> is used.
-r	region size	Defines the region for the message using coordinates. It must have four numbers for the rectangle (top, left, height, width). <i>When omitted for the <i>new_msg</i> command the whole display size will be used.</i> <i>When omitted for all other commands, the last defined region size will be re-used if no region size or line flags are provided.</i>
-n X/M	region size	Defines the region for the message using lines. For <i>new_msg</i> , it specifies that the entire display face will be treated as M lines, and that the current command should operate on line X of those. For <i>add_region</i> and <i>add_text</i> , this specifies that the region occupied by the message is divided into M lines, and line X will be used. The lines are 'one-based', ranging from 1 through M.
--		Indicates the beginning of the positional parameters for the command specified in -c

Table 1: Command Line Flags

The command flag (-c) is always required. The other flags are optional.

The region size (-r) and line-based size (-n X/M) can be used with any of the commands. However:

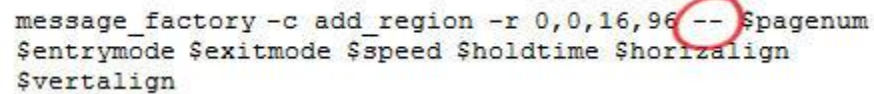
- Before you can refer to a given region size or line-based region with the *add_text* command, you must create that sized region with the *add_region* command.
- If you specify a region size or line-based region with the *new_msg* command that is smaller than the overall size of the display, you cannot specify regions outside of this region with the *add_region* or *add_text* commands even if they would fit inside your display.

CONFIDENTIAL

Note that the positional parameters needed for each of the different commands will vary. These parameters are positional because they must be provided in the order defined by the command. Most commands have a set of required positional parameters, followed by additional optional positional parameters. Refer to Section 6.4 for specific parameters of each command.

6.3.1. About the "--"

It is a good practice to separate the positional parameters of the command from its flags with a --, as in Figure 4.



```
message_factory -c add_region -r 0,0,16,96 -- $pagenum
$entrymode $exitmode $speed $holdtime $horizalign
$verticalign
```

Figure 4: Using the --

This helps to make it more obvious that the positional parameters must be placed in the exact order defined even though the flags are not order dependent.

6.3.2. Error Handling Notes

The message_factory commands are intended to be used by automated scripts and expert users. Thus the message factory itself only performs a small set of command validation checks. Most validation is performed when the message recipe file is used to play the message with the localmsgs command.

During the conversion of the recipe file into an LLM message file, errors present in the recipe file will be reported to the user. The message_factory does perform the following checks:

- If the recipe file does not exist, the first command must be *new_msg*.
- The line abstraction (i.e., the -n flag) cannot specify a number of lines greater than is supported by the display size.
- The line abstraction cannot specify a line number greater than the total lines.
- The line abstraction cannot use a line smaller than 8 pixels high.

6.4. Positional Parameters for Individual Commands

6.4.1. Parameters for Starting a New Recipe (new_msg)

The *new_msg* command will delete the recipe file specified if a file with that name already exists. A new file will be started with the given priority or the default of

CONFIDENTIAL

NORMAL priority if none was specified. All of the other commands require *new_msg* to be invoked before they can operate on a recipe file.

Parameter	Explanation		Required?
Named priority	One of the defined priority names:		✗
	Background	Typically used for the Time and Date Message, lowest priority	
	Normal	Default priority of messages created with the localmsgs scripting command	
	High, Urgent, Emergency	Available priorities for messages above normal priority	
	Critical	Typically used by Inova OnAlert Server to convey high priority emergency messages	
	Panel	Reserved for messages received from a Life Safety system such as a Fire Panel	

Table 2: *new_msg* Parameters

One of the options for the compose commands is *-y priority*. These values are listed in Table 2 in inverse order of precedence (i.e., *Panel* is the highest priority and *Background* is the lowest).

Of the messages in the Local Folder, only those of the highest present priority will ever be visible; these messages will appear in round robin style on the display.

Messages of lower priority will not appear on the display face; their status will always remain “Waiting” until all messages of higher priority are canceled.

6.4.2. Parameters for Adding Message Regions (*add_region*)

Parameter	Explanation	Required?
Page number	Page on which to add the region integer. The default is 1.	✓
Entry mode	appear, ribbon_left, ribbon_right, scroll_up, scroll_down, fade, slide, spell, venetian, expand, contract	✓
Exit mode	appear, ribbon_left, ribbon_right, scroll_up, scroll_down, fade, slide, spell, venetian, expand, contract	✓
Speeds	slowest, slow, medium, fast, fastest	✓
Hold time	0 to 3600000 in milliseconds This is required for appear modes. For ribbon and scroll modes it causes the message to pause in the middle of play.	✓
Horizontal alignment	left, center, right	✓

CONFIDENTIAL

Vertical alignment	top, middle, bottom	✓
Sound name (optional)	beep, staccato, first_call, peter_gunn_theme, fifth_symphony, toccata_and_fugue, charge, octave_up, octave_down, shrill, increasing, decreasing, happy_birthday	✗
Repetition for sound (optional)	1 to 20	✗
Output mode - for sound (optional)	never, activation, appearance Activation only plays the sound the first time the message plays, while appearance plays every time the message repeats.	✗

Table 3: *add_region* Parameters

6.4.3. Parameters for Adding Text (*add_text*)

Parameter	Explanation	Required?
Page number	page on which to add the text	✓
Font size	values between 6 and 32 are accepted, but not all exist as valid fonts	✓
Font style	normal, bold, wide	✓
Font family	block, profile, symbol	✓
Font spacing	none, condensed, normal, wide	✓
Text color	black, red, green, yellow, vrainbow, hrainbow	✓
Background color	black, red, green, yellow	✓
Blink style	none, black, inverse, rainbow	✓
Blink rate	none, slow, medium, fast	✓
Display text	the text to add to the display region, enclosed in single quotes - OR - the hex code string for symbol	✓

Table 4: *add_text* Parameters

6.4.4. Parameters for Adding Data Fields (*add_df*)

Parameter	Explanation	Required?
Page number	page on which to add the data field	✓
Font size	values between 6 and 32 are accepted, but not all exist as valid fonts	✓

CONFIDENTIAL

Font style	normal, bold, wide		✓
Font family	block, profile, symbol		✓
Font spacing	none, condensed, normal, wide		✓
Text color	black, red, green, yellow, vrainbow, hrainbow		✓
Background color	black, red, green, yellow		✓
Blink style	none, black, inverse, rainbow		✓
Blink rate	none, slow, medium, fast		✓
DF description	text to describe the data field		✓
DF type	UNKNOWN, INT16, UINT16, INT32, UINT32, DOUBLE, CHAR, STRING, BOOL, DATE, TIME, DATETIME, REALTIME		✓
DF ID	The Data Field ID in format server,source,field 65534,65534,1		✓
DF format	Int16	INT16, LONG, CURRENCY, FLT_1, FLT_2, FLT_3, FLT_4	✓
	Uint16	INT16, LONG, CURRENCY, FLT_1, FLT_2, FLT_3, FLT_4	
	Int32	INT16, LONG, CURRENCY, FLT_1, FLT_2, FLT_3, FLT_4	
	Uint32	INT16, LONG, CURRENCY, FLT_1, FLT_2, FLT_3, FLT_4	
	Double	INT16, LONG, CURRENCY, FLT_1, FLT_2, FLT_3, FLT_4	
	Char	TEXT	
	String	TEXT	
	Bool	LOGICAL, LOGICAL_CAPS	
	Date	MM_DD_YY, MM_DD_YYYY, MONTH_DD_YYYY, MON_DD_YYYY, DAY_MONTH_DD_YYYY, DD_MM_YY, DD_MM_YYYY	
	Time	12_HH_MM_SS, 12_HH_MM, 12_HH_MM_SS_AM, 12_HH_MM_AM, 24_HH_MM_SS, 24_HH_MM, TM_IN_SEC, TM_IN_MIN	
	Datetime	12_HH_MM_SS, 12_HH_MM, 12_HH_MM_SS_AM, 12_HH_MM_AM, 24_HH_MM_SS, 24_HH_MM, MM_DD_YY, MM_DD_YYYY, MONTH_DD_YYYY, MON_DD_YYYY, DAY_MONTH_DD_YYYY,	

CONFIDENTIAL

		12_HH_MM_AM_MM_DD_YY, 12_HH_MM_SS_AM_MM_DD_YYYY, DD_MM_YY, DD_MM_YYYY	
	Realtime (System Date / Time)	12_HH_MM_SS, 12_HH_MM, 12_HH_MM_SS_AM, 12_HH_MM_AM, 24_HH_MM_SS, 24_HH_MM, MM_DD_YY, MM_DD_YYYY, MONTH_DD_YYYY, MON_DD_YYYY, DAY_MONTH_DD_YYYY, 12_HH_MM_AM_MM_DD_YY, 12_HH_MM_SS_AM_MM_DD_YYYY, DD_MM_YY, DD_MM_YYYY	
DF alignment	ALIGNL, ALIGNC, ALIGNR		✓
DF width	Field Width, 0 = Use Default Min = 1, Max = 4000		✓
DF thresh (repeatable)	Threshold Value, Min = 0 Max = 32767		✗
Text color (repeatable)	black, red, green, yellow, vrainbow, hrainbow		✗
Background color (repeatable)	black, red, green, yellow		✗
Blink style (repeatable)	none, black, inverse, rainbow		✗
Blink rate (repeatable)	none, slow, medium, fast		✗

Table 5: *add_df* Parameters

CONFIDENTIAL

All claims based on information publicly available at time of printing. All other product or service names mentioned in this document may be trademarks of the companies with which they are associated.

© 2010 Inova Solutions. | All rights reserved. 6.17.2011 | page 21

6.5. Two-line Example Message with Time Date Field

The example in Figure 5 constructs a recipe file for a message that has text on line one and the current wall clock time (as a real-time field) on line two.

```
message_factory -c new_msg -r 0,0,16,96 -- high

message_factory -c add_region -n 1/2 -- 1 appear appear fast
5000 left bottom

message_factory -c add_text -- 1 8 normal profile normal
yellow black none none 'current time'

message_factory -c add_region -n 2/2 -- 1 appear appear fast
5000 left bottom

message_factory -c add_df -- 1 8 normal profile normal green
black none none "realtime df" realtime 65534,65534,1
24_HH_MM_SS ALIGNC 0
```

Figure 5: Two Line Sample Message

6.6. Message Status

Use the message status command to check which messages are playing on the display. This command reports the schedule and visibility of the messages playing on the display at the time the msgstatus command is issued. To view the message status, enter `localmsgs msgstatus`. This will issue a report of the messages playing on the display. In the sample message status report (see Figure 6), four messages are in the localmsgs folder, or four messages are “playing” on the display.

OnAlert Display Local Folder Manager					
Local Folder Message Status :					
Message Name	Showing	Schedule	State	Priority	Schedule
__default_time.llm	Waiting	Active		Background	Start now; end never
mymsg.llm	Waiting	Active		High	Start now; end never
myothermsg.llm	Visible	Active		High	Start now; end never
mythirdmsg.llm	Waiting	Active		High	Start now; end never

Figure 6: Message Status Report

An alternative method would be to use a Secure Copy (SCP) command to bring the message status file from the display and parse it on the controlling server, as in the following example, run on the controlling server:

```
scp -pw <passwd> admin@display:/home/admin/localmsgs.status
/local_folder/file
```

CONFIDENTIAL

Note that on an infrequent basis, the status file will be in the process of being updated and a “file not found” error will be generated. Scripts should be written to handle this case; the status file will be available again within a very short time.

Using either the localmsgs msgstatus command or the scp alternative does consume resources on the display. A script that often polls the display for message status may impact the performance of the messages currently playing. It is a good idea to avoid making more frequent status checks than necessary.

CONFIDENTIAL

Appendix A - More on the localmsgs Command

The following help information is provided from the display when the 'localmsgs - h' command is entered.

Usage: localmsgs [command] [options]

Where command is one of the following:

- list [-q] - list the Local Folder messages (default)
- compose [-bcdmpstx] - create a simple message in the Local Folder
- props [-f] - display the properties of a message
- msgstatus [-q] - list the Local Folder messages with current play status
- add [-fS] - add a Local Folder message using a remote tftp server
- delete [-f] - delete Local Folder messages (-f ALL deletes all)
- copy [-fFP] - copy a message into the Local Folder
- rename [-fr] - rename a file in the Local Folder

Where options are one or more of the following:

- -b - select the bold version of the font
- -c color - select the font color (red, green, yellow, vrainbow, hrainbow)
- -d dwell - set the message dwell time in seconds (0-3600)
- -f file - specify a target filename
- -e recipe - use an existing llm_builder "recipe" file to create an LLM message file. The recipe file is deleted afterwards.
- -F folder - specify a folder to copy from (startup, library)
- -h - this help message
- -m melody - play a melody (beep, charge, decreasing, fifth_symphony, first_call, happy_birthday, increasing, octave_down, octave_up, peter_gunn_theme, shrill, staccato, toccata_and_fugue, or none)
- -p mode - presentation mode (appear, ribbon_left, ribbon_right, scroll_up, scroll_down, fade, slide, spell, venetian, expand, contract)
- -P path - specify a path to copy from (instead of a folder)
- -q - quiet mode, no extraneous text
- -r rename - specify a renamed filename

CONFIDENTIAL

- -s size - specify a font size (6, 8, 11, 16)
- -S server - specify a tftp server
- -t text - specify the message text (always enclose in single quotes)

Note that double quotes are converted to single quotes before being displayed on the sign. Use a double quote if you would like a single quote in a message.

For example: -t 'Exit using the "side" door'

Displays on the sign as: Exit using the 'side' door

- -x count - repeat the melody count times (1-20) (defaults to 1)
- -y priority - set message priority (background, normal, high, urgent, emergency, critical, panel)

CONFIDENTIAL

All claims based on information publicly available at time of printing. All other product or service names mentioned in this document may be trademarks of the companies with which they are associated.

© 2010 Inova Solutions. | All rights reserved. 6.17.2011 | page 25