Message is an application for interactively sending *Speaker* (Objective-C interface to Mach messaging) messages to other applications. Familiarity with the documentation for the Speaker/Listener objects as well as Mach messaging in general will help in understanding the use of this application. The top box in the window contains fields to fill out in order to send a message:

Host: The name of the host to receive the message, by default `localhost`. You can enter other host names by typing them in the text field and they will be added to the pop-up list. You can add hosts to be in the pop-up list at startup by adding them to the *machines* directory in NetInfo.

The NetInfo domain used is determined by the *NetInfoDomain* default for

the *Message* application (initially '.', the local domain). If your local and parent domains have too many hosts, you can set this to a dummy value which will cause only *localhost* to be in the list.

You should review the notes in the NXPortFromName section of the NextStep/Reference/03_CFunctions/NXFunctFtoP.rtfd document about messaging remote *Workspace Managers*.

Application: The name of the application to receive the message. You can enter other application names by typing them in the text field and they will be added to the pop-up list.

You can add other applications to be in the pop-up list at startup by setting the *ApplicationPaths* default for the *Message* application to a list of colon (:) separated directories (its default value is `'/NextApps'`).

Port: A port assigned to the recipient application to which the message is sent. By default, the value of this field is *`public`* which sends the message to the application's public port. Other legal values include integers representing ports provided by the remote application.

As with the other fields, you can type in new values into this field that will be added to the pop-up button. You can make additions to the pop-up list at startup by adding named ports to the `ports.strings` file in the

`Message.app` directory.

Message: The message to send to the application. You can add other message names by typing them in the text field and they will be added to the pop-up list. When you hit return, the messages argument fields will be expanded in the *argument box*, awaiting variable *type* assignments.

When the message is actually sent, then the message is added to the pop-up list and the argument types are stored by the program and will be automatically set the next time the message is used. You can add a message to be in the pop-up list at startup by adding it and its argument types to the `messages.strings` file in the `Message.app` directory.

The argument box contains fields that either need to be filled out to send a message (*inputs*) or are the results of a message having been sent (*outputs*). Input fields are white and can be edited; output fields are gray and cannot be changed. The expected *type* of the data is listed in the pop-up list in the center. If you are entering a new message, you will need to set this pop-up list to the appropriate type as well as filling in the arguments.

Byte array arguments are treated like character string arguments for purposes of the user interface. However, the string is followed by its length as a separate argument in the arguments to `selectorRPC:paramTypes:`, excluding the terminal null byte.

For the *send* and *receive* `port_t` types (both inputs), you can also use the special value *`public`* rather than a port number. For the *send* port, public will be replaced with the Listener object's receive port and for the *`receive`* port it will be replaced with the Speaker object's send port.

Once all the fields are filled in, you can send the message by clicking the ***Send!*** button in the bottom box. The result code for the message will be displayed in the ***Status:*** field along with a textual translation of the code if one is known. (Additional result code translations can be added to the `status.strings` file in the `Message.app` directory.) Any output fields in the arguments box will also be filled in based on the results returned by the application.

The ***Message*** application has the following known limitations:

· The contents of various `*.strings` file is sorted (to undo the hash ordering of the `NXStringTable`) when entered into the pop-up list as is the list of applications determined from the directories on the *ApplicationPaths* default. However, the first item in each pop-up list is set by the Message application and is not alphabetical.

· A single Message application can not respond to its own queries. However, if you run more than one Message application, it can.

· Byte arrays entered through the text field cannot have terminal characters (e.g. null bytes) in the middle of them though they are perfectly legal elements of a byte array.