DSNeXTMailer;¬DSNeXTMailer

Inherits From: DSMailer : Object

Declared In: DSNeXTMailer.h

Class Description

The DSNeXTMailer class implements an interface to Mach and NeXTSTEP electronic mail services *without* the need for communication with the NeXTSTEP Mail application.   This is accomplished by directly accessing *sendmail*, the Mach executable that is used to address, format and deliver electronic mail.   The DSNeXTMailer class is also able to generate NeXTMAIL attachments and send electronic mail which contains Rich Text Format body text in a NeXTMAIL compatible format.

This class can be used in one of two ways.   The most easily accomplished approach is to ªattachº it to a Text object that is instantiated in the user interface.   Through the **textFromText:**;DSNeXTMailer.rtf;textFromText:;¬ method, the entire content of the Text object can be converted into a NeXTMAIL message and delivered with very little work (the recipient of the message and the subject must also be set).

Alternatively, the **append:**;DSMailer.rtf;append:;¬ and **appendLiteral:**;DSMailer.rtf;appendLiteral:;¬ methods can be used to create a message from scratch. This approach has the advantage of not requiring any user interface whatsoever to send the message. For standard mail or when no user interface is involved, this can often be the most direct approach. Also note that the DSNeXTMailer can create a message from *any* Text object, not necessarily one which appears in the user interface. Following is an example of using this approach:

```
DSNeXTMailer mailer = [[DSNeXTMailer allocFromZone:[self zone]] init];

[mailer setCC:"sherry"];
[mailer setTo:"someone@dolphin.com"];
[mailer setReplyTo:"postmaster"];
[mailer makeRich:YES];

[mailer appendLiteral:"Dear someone,\n\nEnclosed is the file you requested: "];
[mailer attachment:"/LocalLibrary/Files/InfoFolder"];
[mailer appendLiteral:"\nThank you for using our mail server."];
[mailer deliver];
```

In the above example, a DSNeXTMailer instance is created and ªinitializedº to send mail to ªsomeone@dolphin.comº with a copy to ªsherry,º and replies to ªpostmaster.º It is usually a good idea to provide a reply-to address, as programmatic mail can become lost if it is incorrectly addressedÐand the default return may very well be a rarely used account. The mail is made rich, which is a manditory step when creating NeXTMAIL. Some text is inserted into the message (this text will appear in the default font, which is Times Roman 14 pointÐthe NeXTMAIL default). An attachment is inserted into the stream of text using the **attachment:**;DSNeXTMailer.rtf;attachment:;¬ method; the attachment is a folder, although any file can be inserted. Some additional text is written, and finally the message is delivered.

Instance Variables

BOOL **isNeXTMAIL;**

isNeXTMAIL	Flag which indicates if the message is to be sent as NeXTMAIL

Method Types

Initializing and freeing	;DSNeXTMailer.rtf;init;¬	± init
Message sources	;DSNeXTMailer.rtf;append:;¬	± append:
	;DSNeXTMailer.rtf;attachment:;¬	± attachment:
	;DSNeXTMailer.rtf;textFromText:;¬	± textFromText:
Message actions	;DSNeXTMailer.rtf;deliver;¬	± deliver
	;DSNeXTMailer.rtf;makeRich:;¬	± makeRich:

Instance Methods

append:;¬append:
± append : (id) *anObject*

Appends the string value of *anObject* to the message body. The instance *anObject* must support the **stringValue** method (the DSStream and all subclasses of the DSMemory class support the method). This can be used to ªincrementallyº build a message. The DSNeXTMailer implementation of this method varies from the superclass method only in that it must provide some special handling for NeXTMAIL messages. Return **self**.

attachment:;¬attachment:
± attachment : (const char *) *attachment*

Creates an attachment and appends the attachment to the end of the message. The string pointed to by *attachment* must point to a valid directory or file which has read permission established for the process sending the message. Using this method will automatically call **makeRich:** if it has not already been done. Returns **self***.*

deliver;¬deliver
± (BOOL) **deliver**

Sends the message by invoking *sendmail* and piping in the message body. If **makeRich:** or **textFromText:** were used to create the message, the resulting electronic mail will conform to NeXTMAIL format. Otherwise, the message will conform to Mach mail standards and contain no attachments or Rich Text Formatting. Returns a boolean indication of success. If NO is returned, there was a problem delivering the messageÐthis could be due to improper addressing or the lack of a ªTo:º field, or due to *sendmail* failure on the mail server.

Note that the delivery process for a NeXTMAIL message requires that file attachments be temporarily copiedÐtherefore, there must be enough disk space available to momentarily duplicate the attachments in the message.

See also: ± makeRich:;DSNeXTMailer.rtf;makeRich:;¬, ± textFromText:;DSNeXTMailer.rtf;textFromText:;¬

init;¬init
± init

Returns a newly initialized DSNeXTMailer instance. This method should be sent after the **alloc** method. A newly created DSNeXTMailer assumes that *sendmail* is located in /usr/lib. Returns **self***.*

makeRich:;¬makeRich:
  ± makeRich : (BOOL) *flag*

Makes the message rich text (RTF) or not, depending on *flag.* If a message is currently in rich format when this message is received and *flag* is NO, information will be lost. Attachments and rich text format messages must be made rich. Methods implemented in the superclass, such as **textFromString:**;DSMailer.rtf;textFromString:;¬, will *not* automatically make the message rich if they contain RTF formatting. Returns **self***.*

Note: The current implementation of the DSNeXTMailer does *not* support the ability to convert an RTF message into a non-RTF message. Once **makeRich:** has been received with an argument of YES, the current message cannot revert to standard non-RTF mail.

See also: ± textFromText:;DSMailer.rtf;textFromText:;¬

textFromText:;¬textFromText:
  ± (BOOL) **textFromText :** (Text *) *aTextInstance*

Creates a message from the contents of the Text instance, *aTextInstance*. As with **textFromString:**, this method will erase any other text already contained in the message. Messages created with this approach will be delivered as NeXTMAIL, even if no attachments or explicit text formatting have been used. The receiver will automatically call **makeRich:** when this method is received.

Generating NeXTMAIL from a NeXTSTEP application is very easy. It can be accomplished with a few lines of code, shown below. Given a Text object in the user interface, *theText*, and an already instantiated DSNeXTMailer object, *mailer*, the Text object should be prepared to accept formatting and attachments as follows:

```
[theText setMonoFont:NO];
[theText setFontPanelEnabled:YES];
[theText setGraphicsImportEnabled:YES];
```

This will enable use of rulers, full rich text formatting, and file attachments by dragging and dropping. Once the text and attachments of the message have been supplied by the application user, the following lines of code will address and deliver the message:

```
[mailer setTo:[toField stringValue]];
[mailer setSubject:[subjectField stringValue]];
[mailer textFromText:theText];
[mailer deliver];
```

If the Text instance contains any attachments these attachments will be converted from RTFD format into NeXTMAIL format, bundled into the message, and delivered. Note that enough free disk space must be available in the temporary directory to momentarily copy all of the enclosed attachments. Returns a boolean indication of success.

See also: ± makeRich:;DSMailer.rtf;makeRich:;¬, ± attachment:;DSNeXTMailer.rtf;attachment:;¬, ± textFromString:;DSMailer.rtf;textFromString:;¬