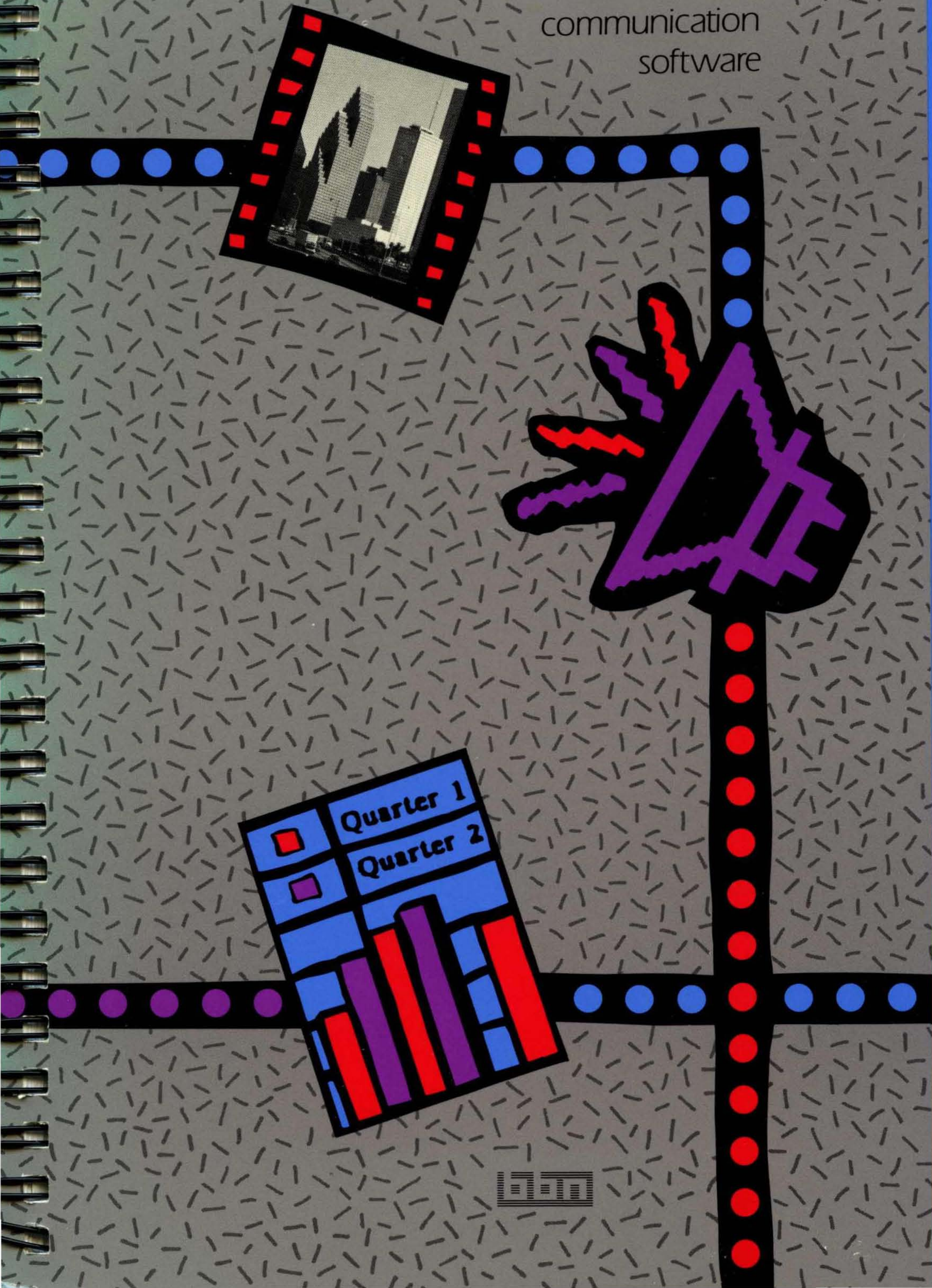
multimedia
document
communication
software

BBN/Slate™



System Topics

multimedia
document
communication
software

BBN/Slate™

System Topics

This document reflects the BBN/Slate Release 1.1 software.

BBN Software Products
A Division of Bolt Beranek and Newman Inc.

BBN Software Products
A Division of Bolt Beranek and Newman Inc.
150 CambridgePark Drive
Cambridge, MA 02140

Latest Printing: October 1990

Contents

Preface

This preface contains the following:

- an overview of this manual
- a list of the documentation conventions used in this manual
- an overview of the BBN/Slate™ documentation

About This Manual

System Topics discusses system-related subjects such as the structure of BBN/Slate document files, command-line document management tools, and mail utility programs. This manual also includes information for system administrators about the site configuration of the BBN/Slate software. This manual contains six chapters:

- Chapter 1 describes the configuration and data files used by different components of the BBN/Slate system.
- Chapter 2 explains the document format used by BBN/Slate. The chapter covers the general document architecture as well as the specific format currently in use.
- Chapter 3 discusses the advanced mail handling features of the MMDeliver mail program.
- Chapter 4 describes the MMail program, which you can use to read and manage BBN/Slate documents from an ordinary computer terminal rather than from a workstation.
- Chapter 5 describes how to use the MMScan image-scanning program on a personal computer to scan images for BBN/Slate documents.

- Chapter 6 presents the BBN/Slate command-line tools available for managing collections of documents at the UNIX shell level.

Documentation Conventions for This Manual

Unless otherwise noted in the text, this book uses the conventions described in this section.

Mouse Operations

This book uses the following format for describing mouse operations:

BUTTON action

or

modifier BUTTON action

where:

- **BUTTON** is the left, middle, or right button.
- **Action** is click, hold, drag, or release.
- **Modifier** is the key you hold down while performing a mouse operation, either the control or shift key.

For example:

LEFT click means you click the left mouse button.

Shift MIDDLE hold means you press and hold the middle mouse button while holding down the shift key on the keyboard.

Menu Commands

Menu commands are given with the form:

menuname-commandname

For example, the following menu command:

Text-Fonts-Face-Italics

shows that **Text** is the top-level menu, **Fonts** is a submenu of **Text**, **Face** is a submenu of **Fonts**, and **Italics** is an option of the **Face** menu.

Boldface Font

This boldface font is used for menu activities and mouse operations. For example:

> You can adjust the spacing of points in the grid with the **Graphics-Style-Grid** command.
>
> To listen to a speech passage in a document, you **LEFT click** while the mouse pointer is over the icon or passage.

Boldface is also used for keys you press on the keyboard, such as the **Return** key.

Italic Font

This italic font is used to give emphasis, especially when a new term is introduced in the text. In discussions of command-line activity or programming, italic represents a placeholder in command syntaxes and directory specifications. For example, in the following command syntax:

```
% install/install_lang package
```

package is a placeholder for the actual name of the language package you are installing.

Monospace Font

`This monospace font` is primarily used in descriptions of command-line activity and programming. It is used to represent filenames, directories, BBN/Slate commands and functions, UNIX operating-system commands, prompts and messages you see on your screen, and user input. For example:

You must define the program by adding it with an `autoload` statement to your `.slate_editor.init` file.

BBN/Slate folder names are also represented in monospace font.

Control Key

For key sequences that use the control key, a caret (^) indicates the control key. For example, **^A** means that you type the control key and the letter A together. You can type A in upper- or lowercase.

Escape Key

The escape key is represented by **Esc**, as in **Esc-D**. In this example, you type the escape key and an uppercase D together. You would not type a lowercase d; case matters when you are using the escape key.

Alternate Key

Key sequences that use the alternate key represent the key with **Alt**, as in **Alt-w**. In this example, you type the alternate key together with lowercase w. Again, case matters when you use **Alt**. Please see the *Release Notes* for information about which key serves as the **Alt** key on your system.

Overview of BBN/Slate Documentation

User Documentation

The BBN/Slate documentation set includes the following manuals:

- *Getting Started* introduces you to BBN/Slate through a series of tutorial exercises.
- The *Reference Manual* gives a complete description of the BBN/Slate system.
- The *Customizing Manual* explains how you can tailor your BBN/Slate system to fit your needs. Topics include reconfiguring the menu system, and writing your own programs using the Slate Extension Language (SEL).

- *System Topics* is a collection of documents covering system topics such as the structure of BBN/Slate document files, command-line document management tools, and mail utility programs.
- *Multilingual Documents* describes how to create documents in five other writing systems (Arabic, Cyrillic, Hangul, Hebrew, and Thai) available with the BBN/Slate Multilingual Option. *Note*: If you did not order the Multilingual Option, you will not receive this manual.

On-Line Help

In addition to the printed documentation, BBN/Slate has an on-line menu help facility that provides information on all command and menu choices. Also, BBN/Slate provides on-line UNIX manual pages with information about BBN/Slate's command-line programs and utilities.

Installation and Release Notes

System-specific installation instructions and release notes supplement the BBN/Slate user documentation set.

Obtaining Documentation and Services

This section explains how you can:

- order documentation
- obtain technical assistance
- obtain training services

How to Order Documentation

To order additional copies of this manual or any other BBN/Slate manual:

- In North America, mail a completed purchase order to:

  BBN Software Products
  A Division of Bolt Beranek and Newman Inc.
  10 Fawcett Street
  Cambridge, MA 02138
  ATTN: Sales Administration

  For prices and other ordering information, call Sales Administration at 617/873-5115.

- Outside North America, contact the nearest Sales and Support office.

How to Obtain Technical Assistance

For BBN/Slate customers who are on maintenance and need technical assistance:

- In North America, call the Hotline number: 617/873-3968. Hotline hours are Monday through Friday, 8:30 a.m. to 8 p.m. (EST).
- Outside North America, contact the nearest Sales and Support office.

How to Obtain Training

BBN supports its customers and products with a full range of training programs. BBN/Slate courses are offered in conjunction with hardware vendors or at BBN education centers. In addition, arrangements can be made with the Education Services department to present the same courses at customer sites.

To receive a course catalog or additional information:

- In North America, call the Course Registrar at 617/873-8383.
- Outside North America, contact the nearest Sales and Support office.

1 Files Used by BBN/Slate

This chapter describes the configuration and data files used by different components of the BBN/Slate system. Some of these files are configuration files that you may edit, while others are data files maintained by the BBN/Slate programs themselves.

The purpose of these descriptions is to give you an understanding of how BBN/Slate is configured, and how the various components of BBN/Slate share information with one another.

The topics covered in this chapter are:

- sitewide configuration files (Section 1.1)
- files used by the Document Editor (Section 1.2)
- files used by the Document Manager (Section 1.3)
- files used by the Document Previewer (Section 1.4)
- files used by the conferencing system (Section 1.5)
- files used by the spelling checker (Section 1.6)
- files shared by all document management programs (Section 1.7)
- font configuration files (Section 1.8)

The filenames in this chapter use the convention `~/filename` to indicate that `filename` resides in a user's home directory. For example, a filename `/usr/slate/lib/slate_editor.init` means the file is in the `lib` subdirectory of the BBN/Slate installation tree, which is usually `/usr/slate`. However, the filename `~.slate_editor.init` means the file is located in a user's home directory. There are a few exceptions, as noted in the descriptions that follow.

1.1 Location of Sitewide Configuration Files

BBN/Slate searches for its sitewide configuration files using the rules listed below. The rules are tried in this sequence either until the files are found, or all the search rules have been tried. The rules used to find sitewide configuration files are:

1. If the environment variable `SLATE_DIR` is set, BBN/Slate looks for sitewide configuration files in `$SLATE_DIR/lib`.
2. If `SLATE_DIR` is not set, BBN/Slate looks for sitewide configuration files in `/usr/slate/lib`.
3. If files are not found in `/usr/slate/lib`, it looks in `/usr/local/lib`.
4. If the files are not found in `/usr/local/lib`, it looks in `/usr/lib`.
5. Finally, if the files are not found in `/usr/lib`, it assumes that there is no sitewide configuration information available.

If other machines are set up to access the BBN/Slate software via NFS, you must make sure those machines also have access to the sitewide configuration files, either via NFS or by copying the configuration files to each machine that mounts the BBN/Slate software directories.

In some cases, you may want different hosts to have different configuration settings. For instance, a collection of hosts that shares one copy of BBN/Slate via NFS may require different mail or print commands on different machines.

In this case, you should make sure that the configuration files are *not* shared among machines, and that each machine's copy of `slate_editor.init` contains the variable settings necessary for that particular machine.

One way you can ensure that the configuration files are appropriate for each machine is to move them into a non-shared directory, then create symbolic links back in their original locations that point to the non-shared directory. The symbolic link will be evaluated on each machine and will pick up the local copy.

1.2 Files Used by the Document Editor

This section describes the files used by the multimedia Document Editor. In addition to these files, the Document Editor uses the font configuration files described in Section 1.8.

`/usr/slate/lib/slate_editor.init`
This is the sitewide editor configuration file. It can contain default variable assignments, keyboard bindings, and mail aliases for all users of one copy of BBN/Slate. The format of the editor configuration file is described in detail in the *Customizing Manual.*

The sitewide editor configuration file is typically used to select an appropriate print spooler and mail system as described in the *Installation Instructions.*

The sitewide editor configuration file is normally managed by your system administrator.

`~/.slate_editor.init`
This is the private or per-user multimedia editor configuration file. It contains each user's private variable assignments, keyboard bindings, and mail aliases. The format is described in the *Customizing Manual.* That manual also lists all of the variables that affect the editor's behavior. The commands that you can bind to the keyboard are listed in the *Customizing Manual* and at the end of each chapter in the *Reference Manual.*

The private configuration file is loaded *after* the sitewide configuration file is loaded, so that individual users can override any of the sitewide defaults. The file includes variable settings to control how multimedia mail is sent, how documents are printed, and whether the document classification features of BBN/Slate are enabled. You can add to your own configuration file to tailor BBN/Slate to your particular requirements.

`/usr/slate/lib/.slate_template`
This is the default document template used to initialize new documents in BBN/Slate if no other template applies. This template file should contain a legal BBN/Slate multimedia document, whose formatting styles and content (if any) will be copied into each empty document created by the editor.

You can specify some other default template either through a private `.slate_template` file stored in your home directory or by setting the editor's `default-document-template` variable. This variable can also be set in the sitewide editor configuration file to change the default template for all BBN/Slate users.

`~/.slate_template`
This is the private default document template, which overrides the sitewide template previously described. Like the sitewide template, this file should contain a legal BBN/Slate document, whose formatting styles and content will be copied into each empty document created by the editor.

If you set the `default-document-template` variable in the one of the editor configuration files, it overrides both the sitewide and private `.slate_template` files.

`/usr/slate/lib/slate_mail.addrs`
This is the sitewide mail address configuration file. It lists the addresses to which multimedia mail can be sent, and the

addresses to which multimedia mail should not be sent. Refer to Chapter 12 of the *Reference Manual* for complete information on the BBN/Slate electronic mail system and how it determines the appropriate message format when sending mail.

This mail address configuration file contains a series of lines, with one address per line.

If an address is listed in the configuration file, it means that multimedia mail can be sent to that address, *unless* the address is preceded by an exclamation point (!). A line that begins with an exclamation point means that multimedia mail should *never* be sent to that address. When listing addresses in this file, you can use an asterisk (*) as a wildcard to match any string of characters.

The following examples are lines from a mail address configuration file:

mlandau@slate.bbn.com
Send multimedia mail to the user named `mlandau` at the host named `slate.bbn.com`.

titipu!town-band
Always send multimedia mail to `town-band` on the host named `titipu`. Notice that you can use UUCP addresses in the address configuration file, if that is what you would normally type in the **To:** field of a message.

***@slate.bbn.com**
Send multimedia mail to any user at the host named `slate.bbn.com`.

!postmaster@slate.bbn.com
Never send multimedia mail to the user named `postmaster` at the host named `slate.bbn.com`. You could combine this with the previous example to mean "send multimedia mail to everyone at `slate.bbn.com` *except* the `postmaster`."

!slate.bbn.com!postmaster
The same as the previous example, but using UUCP syntax. Note that the initial exclamation point is *not* part of the address, but is the character that means "*don't* send multimedia mail."

mlandau@*.bbn.com
Send multimedia mail to the user named `mlandau` at any host whose name ends in `.bbn.com`.

***@*.bbn.com**
Send multimedia mail to any user at any host whose name ends in `.bbn.com`.

!root@*
Never send multimedia mail to a user named `root` on any host.

`~/.slate_mail.addrs`

This is the private mail address configuration file. Its format is exactly the same as the sitewide address configuration file described earlier.

The BBN/Slate Document Manager helps you keep this file up-to-date automatically by checking your incoming messages to see if a new sender has sent you multimedia mail. If it finds that someone new has sent you multimedia mail, the Document Manager offers to add that user's address to your private mail configuration file.

Refer to Chapters 3 and 12 in the *Reference Manual* for more information about the Document Manager and electronic mail.

`/usr/slate/lib/slate_prolog.ps`

This file contains a collection of PostScript definitions that is prepended to the PostScript representation of each BBN/Slate document before it is printed. This prolog file defines certain keywords and graphics operators that are used in the PostScript code generated by the Document Editor.

You should never have to modify this file.

`~/.slate_prolog.ps`

This is a private version of the PostScript prolog file used when printing documents. It is provided for testing and debugging new PostScript operators that may be used by the editor.

We advise you *not* to create another private version of the PostScript prolog file. A private version is never needed in normal operation of the BBN/Slate system, but an incorrect PostScript prolog file can prevent you from printing documents.

`/usr/slate/lib/slate_editor.startup`

This is the startup screen for the Document Editor. This file contains an image in the format used by the image editor, and displays in the Document Editor window when the editor is initializing itself.

`~/.slate_clip`

This is one of three clipboard files used by the **Cut**, **Copy**, and **Paste** commands in the Document Editor. It contains the media elements that you last cut or copied from any document.

If you are running multiple copies of the editor at the same time, they all share the same clipboard, so that you can cut from one editor window and paste into a different one.

The media elements on the clipboard are surrounded by the necessary framework to make them into complete document objects. For example, if you cut a single feature from a graphics element, the **Cut** command will create a completely new graphics object containing that single feature, then put the whole graphics element on the clipboard.

`~/.slate_tbls`

This is the second file used by the BBN/Slate clipboard. It contains the document formatting styles and tables for the last document from which an object was cut or copied. This information is required when pasting information back into a document, to ensure that text styles, colors, patterns, and so on, are correctly preserved.

The styles and tables saved in this file include text styles, counter styles, font tables, texture tables, and color tables. Concatenating this file and the `.slate_clip` file produces a legal BBN/Slate document containing the last information that was cut or copied.

For efficiency reasons, this file is only updated when necessary. If you cut two objects in succession from the same document, the tables are saved during the first cut command, so they do not need to be saved again during the second cut.

`~/.slate_lock`

This is the third BBN/Slate clipboard file. It contains the UNIX process id of the last Document Editor that wrote to the clipboard, a new line, and the name of the document buffer from which information was last cut or copied.

The editor uses this file to determine the ownership of information on the clipboard, so that it can perform **Paste** commands as efficiently as possible.

When pasting information that was cut from a different document, the editor must retrieve the text styles and tables that pertain to the material on the clipboard, then merge them into the tables in the document where the material is being pasted.

When cutting and pasting within a single document, the styles and tables of the document will apply to the material on the clipboard as well, so the editor can omit this extra step.

`/tmp/mmshelf`

This file is used by the **Text-Edit-Copy Selection** command to interact with the SunView text clipboard. When you select **Copy Selection**, the text characters in the selected region of the document are written out to this file. If you then invoke the SunView `Get` or `Paste` command in some other window (like a shelltool), the contents of this file will be copied into that window.

The **Copy Selection** command does not try to preserve the format of the text it saves, nor does it save the non-text elements of a document.

1.3 Files Used by the Document Manager

This section describes the files used by the Document Manager. In addition to the files mentioned here, the Document Manager uses the files described in Sections 1.7 and 1.8.

`/usr/slate/lib/slate_tool.init`

This is the sitewide Document Manager configuration file. It contains variable assignments similar to those used in the editor configuration file. Key bindings are not currently supported in the Document Manager.

This file is normally required only in cases where BBN/Slate is not installed in `/usr/slate`. In such cases, the sitewide configuration file must include a line of the form:

```
SLATE_DIR = "Slate installation directory";
```

so that the Document Manager can find the fonts, document templates, and so on, that it needs.

`~/.slate_tool.init`
This is the per-user Document Manager configuration file. Like the sitewide configuration file, it can contain only variable settings. The only variables that may currently be set in the Document Manager are:

- `default-menu-font`, which sets the font used to display Document Manager menus
- `editor-args`, a string that the Document Manager passes to the Document Editor on its command line when it starts up. You can use this to control the size and position of the Document Editor window by including the appropriate window positioning variables.
- `display-document-size`, an integer which, if set to 1, will cause the size of the document to display at the end of the document summary line.

The private configuration file is loaded *after* the sitewide configuration file is loaded, so that individual users can override any of the sitewide defaults.

1.4 Files Used by the Document Previewer

This section describes the files used by the multimedia document previewer. In addition to the files mentioned here, the previewer uses the font configuration files described in Section 1.8.

`/usr/slate/lib/slate_preview.init`
This is the sitewide document previewer configuration file. It contains variable assignments similar to those used in the editor configuration file. Key bindings are not supported in the document previewer.

This file is normally required only in cases where BBN/Slate is not installed in `/usr/slate`. In such cases, the sitewide configuration file must include a line of the form:

```
   SLATE_DIR = "Slate installation directory";
```

so that the previewer can find the fonts that it needs.

`~/.slate_preview.init`
This is the private Document Manager configuration file. Like the sitewide configuration file, it can contain only variable settings. The only variable that may be set in the document previewer is `default-menu-font`, which sets the font used to display the previewer's menus.

The private configuration file is loaded *after* the sitewide configuration file is loaded, so that individual users can override any of the sitewide defaults.

1.5 Files Used by the Conferencing System

`/slate/lib/slate_conf.init`
This is the sitewide configuration file for the multimedia conferencing programs, `mmconf` and `confd`. It contains variable assignments similar to those used in the editor configuration file. Key bindings are now supported in the conferencing system.

This file is required only in cases where BBN/Slate is not installed in `/usr/slate`. In such cases, the sitewide configuration file must include a line of the form:

```
   SLATE_DIR = "Slate installation directory";
```

so that the conferencing software can find the fonts, auxiliary tools, and so on, that it needs.

`~/.slate.conf.init`
This is a private file where you can list aliases for people with whom you confer often. See Chapter 13 of the *Reference Manual* for more information about this file.

1.6 Files Used by the Spelling Checker

`/slate/lib/slate_words.`*`arch`*
This is the sitewide dictionary used by `mmspell`, the editor's spelling checker. The sitewide dictionary is stored in a machine-specific binary format. There is one sitewide dictionary per machine architecture installed.

`~/.slate_words`
This is the private spelling dictionary, which is also read by `mmspell` when checking the spelling of a document. The format of this dictionary is the same as the format used by the public domain `ispell` program posted to the comp.sources.unix newsgroup on Usenet. The descriptions and examples that follow are derived from the `ispell` documentation.

The dictionary consists of a series of words, one word per line, that are considered to be correctly spelled. Each word may also be followed by a slash (/) character and one or more of the following flags, which specify the kind of transformations that are permitted on that word.

In the description of each flag, an ellipsis (...) represents the root word up to its last character, and a pound sign (#) represents any character found at the end of the word.

You can edit this file with a text editor to make new entries, and you can also use the **Leave ...and Add** command in the spelling checker dialog box to add words from a document to the private dictionary. The spelling checker will periodically read and rewrite this file. At that time, it may combine similar words into a single root word plus the appropriate flags.

The flags that may be appended to a word in the spelling dictionary are:

V Controls the addition of "ive" to the end of a word, often used to turn a verb into an adjective or adverb.

```
...E -> ...IVE (create -> creative)
...# -> ...#IVE (prevent -> preventive)
```

N Controls the addition of "ion", "ication", or "en" to the end of a word.

```
...E -> ...ION (create -> creation)
...Y -> ...ICATION (multiply -> multiplication)
...# -> ...#EN (weak -> weaken)
```

X Controls the addition of "ions", "ications", or "ens" to the end of a word.

```
...E -> ...ION (create -> creations)
...Y -> ...ICATION (multiply -> multiplications)
...# -> ...#ENS (weak -> weakens)
```

H Controls the addition of "th" or "ieth" suffixes, often used to turn a number into an adjective.

```
...Y -> ...IETH (twenty -> twentieth)
...# -> ...#TH (hundred -> hundredth)
```

Y Controls the addition of "ly" to the end of the word, often used to turn an adjective into an adverb.

```
...# -> ...#LY (quick -> quickly)
```

G Controls the addition of "ing" to the end of a word, often used to turn a verb into its gerund form.

```
...E -> ...ING (file -> filing)
...# -> ...#ING (cross -> crossing)
```

J Controls the addition of "ings" to the end of a word.

```
...E -> ...INGS (file -> filings)
...# -> ...#INGS (cross -> crossings)
```

D Controls the addition of "ed" or "ied" to the end of a word, often used to form the past tense of a verb.

```
...E -> ...ED (create -> created)
...@Y -> ...@IED (imply -> implied), if @ is not a vowel
...# -> ...#ED (cross -> crossed, convey -> conveyed)
```

T Controls the addition of "est" or "iest" to the end of a word, often used to form the superlative.

```
...E -> ...EST (late -> latest)
...@Y -> ...@IEST (dirty -> dirtiest), if @ is not a vowel
...# -> ...#EST (small -> smallest, grey -> greyest)
```

R Controls the addition of "er" or "ier" to the end of a word.

```
...E -> ...ER (skate -> skater)
...@Y-> ...@IER (multiply -> multiplier), if @ not a vowel
...# -> ...#ER (build -> builder, convey -> conveyer)
```

Z Controls the addition of "ers" or "iers" to the end of a word.

```
...E -> ...ERS (skate -> skaters)
...@Y->...@IERS (multiply-> multipliers) if @ not a vowel
...# -> ...#ERS (build -> builders, convey -> conveyers)
```

S Controls the addition of "s", "es", or "ies" to the end of a word, often used to form the plural form or a noun, or the third person present tense of a verb.

```
...@Y -> ...@IES (imply -> implies), if @ is not a vowel
...# -> ...#ES (fix -> fixes), if # is h, s, x, or z
...# -> ...#S (bat -> bats, convey -> conveys)
```

P Controls the addition of "ness" or "iness" to the end of a word.

```
...@Y->...@INESS (cloudy -> cloudiness) if @ not a vowel
...# -> ...#NESS (late -> lateness, grey -> greyness)
```

M Controls whether a word can be made possessive by adding 's to the end.

```
...# -> ...#'s (dog -> dog's)
```

1.7 Files Shared By All Document Management Programs

`.slate_docs`

This file maintains a history of the multimedia documents in a directory, and their status information. There is one `.slate_docs` file in each directory that contains BBN/Slate documents.

Various document management programs (such as MMail and the multimedia command-line tools) use this file to quickly access information about each document's headers, size, date of last modification, and status.

The file is created automatically the first time it is referenced by any of the BBN/Slate programs, and is automatically updated whenever a program detects that the directory contents have changed. You should never have to manipulate the `.slate_docs` file by hand.

All information in the `.slate_docs` file is stored in simple ASCII form. The file begins with a single line of header information containing the following items:

1. The version of the file format in use. The version number is an integer that is right-aligned in a 6-character field. The current file format version is "2".
2. The date and time the file was last updated. The date and time is stored in the canonical UNIX format (the number of seconds since midnight, January 1, 1970). This number is right-aligned in a 10-character field.
3. The number of documents for which this file contains summary information. The document count is an integer that is right-aligned in a 6-character field.

The header line is followed by document summary information for each document in the directory. Each document is described by a number of different fields; each field consists of a name and a value. The document headers comprise one such set of fields. In addition, there are other fields with special meaning to the document management tools.

The summary for a document begins with an integer that indicates how many fields are used to describe the document, followed by a new line. The names and values begin on the line immediately after the field count. The names and values are stored on separate lines in the file, so a document whose field count is *n* will have 2*n* lines of text to describe it.

The general layout of the summary information for a single document is:

```
field count
Filename
name of the file that contains the document
From
from field of headers, or file owner
Date
date field of headers, or last modification date
Subject
subject field of headers, or file name
.
.  Other document headers appear here
.  They are followed by:
.
classify
the index of the highest classification level
    in the document, or -1 if no classification
classify-names
the classification names used in this document,
    stored in the same format as one uses to set
    the classify-names variable in the document
    editor
file-size
the file size in bytes
Status
the document status characters for this document,
    or blank to indicate "normal" status
```

Summary blocks for successive documents follow each other immediately, without any white space between them.

1.8 Font Configuration Files

/usr/slate/lib/slate_fonts.config

This is the sitewide font configuration file for all of the BBN/Slate programs. It lists the languages and fonts installed with your copy of BBN/Slate, and provides a mapping from internal font names to the external files in which those fonts are found.

Font names can be mapped to one font for display and to a different font for printing (for example, you could have the `timesroman` screen font printing as the built-in Palatino or Bookman font on your laser printer).

The Document Editor reads the font configuration file at startup time to determine which fonts should be displayed in font selection menus and dialog boxes. You can use this file to add additional fonts to your system and the editor will incorporate them automatically. You can also use it to remap font names so that they cause different fonts to be displayed.

Each font is described by a series of lines as shown below:

```
FamilyName font-family
Sizes point-sizes
ScreenNormal font-type filename
PrintNormal print-type font-name
DownloadNormal print-type file-name
ScreenItalic font-type filename
PrintItalic print-type font-name
DownloadItalic print-type file-name
ScreenBold font-type filename
DownloadBold font-type filename
PrintBold print-type font-name
ScreenBoldItalic font-type filename
PrintBoldItalic print-type font-name
DownloadBoldItalic print-type filename
PrintCvt print-type filename
Scripts script-names
```

You can include comments in the font configuration file by starting each comment line with a pound sign (#). Successive font descriptions in the file are separated by one or more blank lines.

The **FamilyName** field supplies the font family name that is used in the Document Editor's menus and dialog boxes. The standard font family names for the basic BBN/Slate system are `timesroman`, `helvetica`, and `tv`.

The **Sizes** field provides a list of the sizes in which the font is available. List multiple sizes separated by spaces.

The **ScreenNormal** field indicates where to find the screen fonts for this font family and size when displaying the "normal" font style.

- The *font-type* is an appropriate type, for example, `X11`. With the X Window System, you can include two different **ScreenNormal** lines to specify different screen fonts for use under X and under your windowing system.
- The *file-name* is the name of the file containing the current screen font. If it begins with a slash character (/), it is considered to be an absolute pathname; otherwise the system searches for the file relative to `/usr/slate/fonts`. If the name contains the string `%d`, the `%d` will be replaced with the font size being displayed.

The **PrintNormal** field indicates what printer font to use when printing the current font family in the "normal" font style.

- The *print-type* indicates the kind of printer font in use. This will normally be `postscript`.
- The *font-name* indicates the name by which the current font is known to the PostScript printer.

The **DownloadNormal** field indicates that the current font must be downloaded to the printer before printing a document that uses it. If the font is a built-in PostScript font, you should omit the **Download Normal** field completely.

- The *`print-type`* indicates the kind of printer font to download. This will normally be `postscript`.
- The *`file-name`* indicates where to find the font to download. If the filename begins with a slash character (/), it is assumed to be an absolute pathname. Otherwise, font files are searched for relative to `/usr/slate/fonts`.

The **`ScreenItalic`**, **`PrintItalic`**, and **`DownloadItalic`** fields are identical to the **`ScreenNormal`**, **`PrintNormal`**, and **`DownloadNormal`** fields, but apply to the italic font face instead of the normal font face.

The **`ScreenBold`**, **`PrintBold`**, and **`DownloadBold`** fields are identical to the **`ScreenNormal`**, **`PrintNormal`**, and **`DownloadNormal`** fields, but apply to the bold font face instead of the normal font face.

The **`ScreenBoldItalic`**, **`PrintBoldItalic`**, and **`DownloadBoldItalic`** fields are identical to the **`ScreenNormal`**, **`PrintNormal`**, and **`DownloadNormal`** fields, but apply to the bold italic font face instead of the normal font face.

The **`PrintCvt`** field is used when the character mappings in the screen font are not the same as the character mappings in the printer font.

- The *`print-type`* indicates the kind of printer font in use. This will normally be `postscript`.
- The *`file-name`* indicates where to find the font conversion information. The font conversion file is normally in `/usr/slate/fonts/cvt`. It contains a single line listing the name of the font, followed by a series of lines of the form:

  ```
  screen-font-index print-font-index
  ```

that map indices in the screen font to indices in the printer font. If a screen font index is not found in the conversion table, it is mapped to the same index in the printer font.

The **Scripts** field lists the character sets that may be rendered using this font. It is used mainly by the multilingual option in BBN/Slate. For the fonts supplied with the basic BBN/Slate system, the value of the **Scripts** field should be the single word ADOBE.

Here is an example of the font configuration information for one of the built-in BBN/Slate fonts:

```
FamilyName timesroman
Sizes 8 10 12 14 18
ScreenNormal X11 timesroman%d
ScreenNormal X11 timesroman%d
PrintNormal postscript Times-Roman
ScreenItalic X11 timesroman%di
ScreenItalic X11 timesroman%di
PrintItalic postscript Times-Italic
ScreenBold X11 timesroman%db
ScreenBold X11 timesroman%db
PrintBold postscript Times-Bold
ScreenBoldItalic X11 timesroman%dbi
ScreenBoldItalic X11 timesroman%dbi
PrintBoldItalic postscript Times-BoldItalic
Scripts ADOBE
```

If you have screen representations of other PostScript fonts available, you can make them available in your documents with configuration file entries such as the following:

```
FamilyName bookman
Sizes 10 12 14 18 24
ScreenNormal X11 /usr/local/psfonts/Bookman-Light.%d
PrintNormal postscript Bookman-Light
ScreenItalic X11 /usr/local/psfonts/Bookman-LightItalic.%d
PrintItalic postscript Bookman-LightItalic
ScreenBold X11 /usr/local/psfonts/Bookman-Demi.%d
PrintBold postscript Bookman-Demi
ScreenBoldItalic X11 /usr/local/psfonts/Bookman-DemiItalic.%d
PrintBoldItalic postscript Bookman-DemiItalic
Scripts ADOBE
```

If you have both screen and downloadable printer versions of a Hebrew font, you might use a configuration file entry such as the following to describe it:

```
FamilyName hebrew
Sizes 12 24
ScreenNormal X11 hebrew.%d
ScreenItalic X11 hebrew.%d
ScreenBold X11 hebrew.%d
ScreenBoldItalic X11 hebrew.%d
PrintNormal postscript Hebrew
PrintItalic postscript Hebrew
PrintBold postscript Hebrew
PrintBoldItalic postscript Hebrew
DownloadNormal postscript postscript/Hebrew.ps
DownloadItalic postscript postscript/Hebrew.ps
DownloadBold postscript postscript/Hebrew.ps
DownloadBoldItalic postscript postscript/Hebrew.ps
PrintCvt postscript hebrew.cvt
Scripts HEBREW
```

(For creating BBN/Slate documents in a non-English writing system, you need the BBN/Slate Multilingual Option. The basic BBN/Slate system does not support multilingual documents.)

`~/.slate_fonts.config`
This is a private version of the font configuration file. The format is identical to the sitewide configuration file described earlier. The private configuration file is loaded *after* the sitewide configuration file, so you can use it to add additional fonts or override the default sitewide font mappings.

2 Document Representation

This chapter describes the document format used by the BBN/Slate system. It describes the general document architecture as well as the specific format currently in use.

This chapter covers the following topics:

- document format concepts (Section 2.1)
- document representation structure (Section 2.2)
- document attributes (Section 2.3)
- shared data tables (Section 2.4)
- flattened object hierarchy (Section 2.5)
- how BBN/Slate encodes each media type (Section 2.6)
- layout algorithm (Section 2.7)

2.1 Concepts

A *medium* is a means of carrying information. The BBN/Slate document format provides a way of combining objects expressed in different media into a single composite object which may be edited, stored, transmitted, printed, and manipulated. The Document Editor provides for the inclusion of text, line-drawing graphics, bitmap images, voice, enclosures, and spreadsheets.

A BBN/Slate document is a hierarchical tree structure with objects of a particular media type at the leaves. Formatting information may be associated with internal nodes of the hierarchy; names associated with this formatting information (*e.g.,* Chapter or Section) may imply some additional semantic information. Beyond these names, no semantic information is associated with the hierarchy.

Across a single document certain information must be shared by multiple objects. This information is specified as *tables* at the start of the document. Entries in tables are normally accessed by an index into the table, though some tables support named entries. Tables are used to reference information used by objects more efficiently. For example, the font of a text passage might be stored as an index into the font table rather than requiring the font family, face, and size to all be specified ''in line''. Tables are also used to gather together data that must be shared between objects, such as text formatting styles.

2.2 Overall Document Representation Structure

A BBN/Slate document is normally stored in a single file. Most information is encoded as ASCII text although the file may contain binary (8-bit) data. The document is made up of three parts:

- document attributes encoded as name-value pairs
- a set of shared data tables
- the flattened object hierarchy

2.3 Document Attributes

Attributes are encoded as ASCII name-value pairs. Some attributes are required to be present while others are only used by a particular media editor or application to store information with a document across editing sessions. These attributes should have reasonable defaults and their absence should be handled easily by the editor or any other application that is manipulating the document. For example, information make about the document format version and the resolution at which it was created is required, but information related to printing is only required when the document is printed and such fields have defaults for page size, margins, and so on. An example of the document attribute section with all the required fields follows:

```
<list>
<protocol bbn-multimedia-format>
<version 2>
<xresolution 75>
<yresolution 75>
<global_font 0>
<page-width 6.5i>
<endlist>
```

The document attribute section begins with the literal token `<list>` and ends with the token `<endlist>`. Each name-value pair begins on a new line. A name that contains any spaces must delimit the space with a backslash (\). The name is followed by a single space and then the text of the value. A new line in the value may be delimited with a backslash. The value terminates with `>\n`.

Resolution is given in pixels-per-inch. The `global-font` is an index into the font delta table (see Section 2.4.6). The `page-width` is the width on the screen, not on the printer (there are other attributes associated with the width of the printed page).

2.4 Document Tables

There are nine kinds of document data tables, which are summarized below. Detailed descriptions follow.

Color The color table stores red, green, blue intensities as 3 8-bit values.

Line	The line table bundles various line attributes together so they can be specified with a single index.
Texture	The texture table stores bit patterns for rendering textures.
Color Texture	The color texture table combines a texture with a foreground and background color to form a colored texture. Both the color and texture are specified as indices into the other tables.
Font Family	The font family table stores font family names, such as `times` or `helvetica`. It is normally only referenced by the font delta table.
Font Delta	A font can be specified as an absolute family, face, and and size, or it can be specified as being relative to, or a *delta* to the font of the surrounding context. This makes it possible to change the font family or size of an entire document while maintaining information about face changes or incremental size changes.
Text Styles	The text style table defines a *style sheet* for formatting the document. Each style has a name and contains information about the font, margins, and the justification that should be used when displaying or printing text formatted by a particular style.
Counters	Counters are used to support numbering of items such as sections, tables, and figures. Counters also support cross references and tables of contents generation.
Tables of Contents	This table defines the kind of table of contents that should be generated for a document.

Each of the tables has a free-format text representation. The general form is:

```
table tabletype
   ...contents...
endtable tabletype
```

Empty tables may be omitted. If the style table for a document is omitted, it is initialized from the default document template.

2.4.1 Color

The general form of the color table is:

```
table color
color red1 green1 blue1 opaqueness1
color red2 green2 blue2 opaqueness2
    •
    •
endtable color
```

The red, green, blue intensities are unsigned 8-bit values. Opaqueness is either `o` for opaque or `t` for transparent.

2.4.2 Line

The general form of the line table is:

```
table line
index fcolor bcolor texture linewidth pattern
    brushtype putline
    •
    •
endtable line
```

The line fields are:

index The index of this entry. Most tables assume an ordinal counting scheme (starting at 0) based on their lexical position in the table. This table explicitly includes the index.

fcolor The foreground color; an index into the color table. The foreground color is the color used for 1 bits in the line texture.

bcolor	The background color; an index into the color table. The background color is the color used for 0 bits in the line texture.
texture	The texture used to draw the line; an index into the texture table.
linewidth	The width of the line in units of the document resolution.
pattern	The pattern used to draw the line; legal values are 0 (solid), 1 (dotted), 2 (dot-dash), 3 (dashed).
brushtype	The style of brush; legal values are 0 (circular) and 1 (angular).

2.4.3 Texture

The texture table is used for specifying bit patterns used to fill areas. The form of the texture table is:

```
table texture
index width height opaqueness hexstring putpat
    .
    .
endtable texture
```

The texture fields are:

index	The index of this entry.
width	The width of the bit pattern.
height	The height of the bit pattern.
opaqueness	A Boolean that indicates if the pattern is opaque or transparent.
hexstring	A Postscript *hexstring* that encodes the bits.

BBN/Slate handles only textures that are eight bits square and allows one white transparent texture. The standard texture table looks like the following example:

```
table texture
0 8 8 false <0000000000000000> putpat
1 8 8 true <0000000000000000> putpat
2 8 8 true <FFFFFFFFFFFFFFFF> putpat
3 8 8 true <FF000000FF000000> putpat
4 8 8 true <8888888888888888> putpat
5 8 8 true <8844221188442211> putpat
6 8 8 true <1122448811224488> putpat
7 8 8 true <FF00FF00FF00FF00> putpat
8 8 8 true <AAAAAAAAAAAAAAAA> putpat
9 8 8 true <AA00800088008000> putpat
10 8 8 true <11A3C78B11B87C7A> putpat
11 8 8 true <031BD8C00C8DB130> putpat
12 8 8 true <77FFDDFF77FFDDFF> putpat
13 8 8 true <55AA55AA55AA55AA> putpat
14 8 8 true <EEBBEEBBEEBBEEBB> putpat
15 8 8 true <88CCEEFF88CCEEFF> putpat
16 8 8 true <010204081C22C180> putpat
endtable texture
```

2.4.4 Color Texture

The color texture table combines a texture with a color for the foreground pixels and a color for the background pixels. This can be used to fill an object with a colored texture. The form of the table is:

```
table colortexture
index nplanes texture forecolor backcolor putctext
endtable colortexture
```

The color texture fields are:

index	The index of this entry.
nplanes	Number of planes of color.

texture	The bitmap texture; an index into the texture table.
forecolor	The color of the foreground (one) bits; an index into the color table.
backcolor	The color of the background (zero) bits; an index into the color table.

(The editor is currently ignoring the color information.)

2.4.5 Font Family

The font family table encodes the names of available fonts. The standard table is:

```
table fontfamily
family timesroman
family helvetica
family tv
endtable fontfamily
```

The `tv` font is a fixed-width font. It maps to Courier for printing.

2.4.6 Font Delta

Within the document, fonts are recorded as deltas to the global document font. This makes it possible to change the font of an entire document with a single operation, while maintaining information such as which words are italic or bold. A font description includes the font family, face, size, an offset from a baseline, some optional information about underlining and overstriking, as well as an optional name. The global font specifies all of these (and is said to be *fully absolute*). Other fonts, such as the font of a paragraph, are normally recorded as relative to this global font.

The format of the font delta table is:

```
table fontdelta
delta: [field = value;]
    .
    .
endtable fontdelta
```

If a field is not specified, it is assumed to be relative. The fields that may be specified are:

fontfamily	The font family to use. This is an index into the font family table.
fontsize	The point size of the font to use. Current point sizes supported in the Document Editor are 8, 10, 12, 14, and 18. If the number begins with an initial plus (+) or minus (-) sign, the size is taken to be relative; otherwise it is absolute.
fontface	The font face to use. Legal values are `roman`, `bold`, `bold-italic`, and `italic`. If specified, the font face is always absolute.
fontbase	The baseline offset in points. This is used for superscripts and subscripts. This value is always assumed to be relative. A negative value moves the baseline *up*.
fontspread	The number of points to add or subtract from the normal width of each character. This can be used to compress or expand text.
fontflags	Other text characteristics, listed below. Each font flag has a value of true or false.
underline	A true value indicates the text should be underlined.
doubleline	A true value indicates the text should be double underlined.
strikeout	A true value indicates the text should be struckout.

textorder	A true value indicates that the text flows right-to-left.
unrender	(Only produced with the Multilingual Option.) A true value indicates that the text should not be explicitly rendered (*e.g.*, to handle ligatures). This has no effect in English text.
fontname	An optional name to associate with this delta. Names are useful for two purposes. One is for associating a particular formatting style with some concept. For example, you may want to define the font delta `command_name` to have a bold face and be one point size less than the surrounding text. The name allows the user access to this formatting information in a meaningful way. In addition, it is possible to redefine the attributes of the font delta with that name and have the appearance of the text change throughout the document. (Not implemented.)
charset	(Only produced with the Multilingual Option.) The character set of the associated characters. Non-English text is represented with a symbolic character set name and then either 8 or 16 bits of actual data per character. Some examples of character set names are HANGUL, HEBREW, CYRILLIC, or ARABIC.
keyboard	(Only produced with the Multilingual Option.) The virtual keyboard that was used to input the associated text. Recording the virtual keyboard supports the ability to change the way keys are interpreted based on the location of the text cursor. Some examples of keyboard names are HANGUL, THAI, and RUSSIAN.

An example of a fontdelta table follows:

```
table fontdelta
delta: fontfamily = 1; fontface = roman;
    fontsize = 10; fontflags underline = false;
    fontflags doubleline = false;
    fontflags strikeout = false;
delta:
delta: fontfamily = 2;
delta: fontface = bold;
delta: fontsize = +2; fontflags underline = true;
    fontname = "command";
endtable fontdelta
```

The initial record indicates that all fields have been specified absolutely. The second record indicates that none of the fields are absolute, meaning that all the attributes are inherited from the surrounding context. Many text styles will use this value to indicate that they simply inherit the document global font. The next record indicates a conversion to bold face. The final record specifies a named font delta that is used for displaying command names.

2.4.7 Text Styles

Text styles are specified in the format table. Styles are used for formatting paragraphs as well as specifying formatting attributes of lists of objects. The form of the format table is:

```
table format
   ...sequence of formats...
endtable format
```

where each format is of the form:

```
format [nonprimal]
   ...sequence of fields...
endformat
```

and a field is of the form:

```
<fieldname> = <value>
```

The first field in each format must be the *name* field, such as:

```
name = "paragraph"
```

The second field must either be the *base* field or the *type* field. The base field is used to allow inheritance of formatting characteristics. Any field that is not explicitly specified in a format is inherited from the format on which it is based. If a format is not based on any other you must specify its type to be either `text` or `list`. For example:

```
   name = "verbatim"
   type = text
or
   name = "enumeration"
   type = list
```

In the default format table distributed with the BBN/Slate, all the text formats are eventually derived from the `verbatim` format. This allows you to change some attributes across the entire document by changing them only in the `verbatim` format. For example, you would add double-spacing to an entire document by simply changing the line spacing in `verbatim`.

Text and list formats have different kinds of attributes associated with them (although there is some overlap).

Certain attributes (*e.g.,* margins) are specified with numeric values. These attributes may be specified in the units listed below:

i	inches
c	centimeters
l	font height (el represents "line height'')
s	width of a space in the current font

p	points (1/72 of an inch)
m	width of an 'm' in the current font

Each attribute assumes a default unit if no unit is explicitly specified.

The attributes associated with a text format are:

help A short descriptive string, explaining the purpose of the format.

font One of two ways that specify a font. A font can be expressed as an index into the fontdelta table:

```
fontindex = 1
```

or it can be expressed symbolically:

```
font =
   fontfamily = "helvetica";
   fontface = bold; fontsize = 10;
```

Any or all of the font attributes may be omitted, indicating that a font attribute is relative to the surrounding font. In addition, the `fontsize` may have a leading plus (+) or minus (-) sign indicating that the size is relative.

lineheight Indicates the spacing between lines. The default units are font heights. For example, 1l (that is, one el) produces single spacing.

indent Amount to indent the first line of the paragraph. Default units are inches.

linewrap A Boolean value that indicates that line breaks are inserted automatically to make the text fit the margins. Boolean values are expressed with the symbols `true` and `false`.

justify	How a line should be placed between the margins. Legal values are `right`, `left`, `center`, and `fill`.
leftmargin	How far to move in the left margin. Default units are inches.
rightmargin	How far to move in the right margin. Default units are inches. The width of a line is determined by subtracting the left and right margins from the default page width.
linewidth	The width of a line from the left margin. Default units are inches. Normally this is zero, indicating that the line width should be derived from the left and right margins and the overall page width. In a few cases (*e.g.*, right-justified list tags) it is important to actually specify the linewidth.
above	White space required before an object. Default units are lines.
below	White space following an object. Default units are lines. The white space below one object and the white space above another overlap so the overall white space is their maximum rather than their sum.
breakpage	A Boolean value that indicates whether a page break should be issued before an object when printing.
keep	A Boolean value that indicates whether to attempt to keep an entire object on a single page when printing.
tabs	The interval to use for placing the default tabs. The default tabs are only used if no special tabs have been defined.
tab	A special tab at a specified position. There may be multiple special tabs in a format. A special tab has the form:

```
tab = <position> <justification> <fill char>
```

`Position` is a normal attribute like "3i" or "40s". *`Justification`* is one of the values `left`, `right`, `center`, or `decimal` to indicate how the text after the tab is supposed to be justified. A `decimal` tab is right justified with the decimal point in the string at the tab stop location. *`Fill char`* is the character to use for filling the space between the text before and after the tab. Typical values are " " (space), "_" (underline), and "." (period).

derivewidth	For layout purposes, indicates whether the width of the object should be determined from the margins or should be determined from the actual content of the object. If `derivewidth` is `true`, *`justification`* must be `left` or `filled`.
nobreakafter	Indicates that a paragraph of this type should not appear at the bottom of the page, but rather should appear on the same page with the paragraph which follows. This is normally used for titles and subtitles.
widow	The number of lines to consider a widow (the number of lines at the bottom of a page). During printing, the formatter will attempt special actions if the default formatting allows this number of lines at the bottom of a page. Default value is 1.
orphan	The number of lines to consider an orphan (the number of lines at the top of a page). During printing, the formatter will attempt special actions if the default formatting allows this number of lines at the top of a page.
content	The default content of a new instance of the current type of object. This is typically used to automatically insert instances of counters when section headings and figure captions are inserted. Text of the form `@counter(name:level:style:value)` is converted into a counter reference; all other text is inserted literally. Octal and C-style escape sequences are interpreted.

valset	A string of comma-separated name-value assignments that are evaluated when an object with this style is printed. This facility can be used to modify the value of page header and footers dynamically during printing. The special name `page` will set the page number. The special value `@` will be interpreted as the current contents of the paragraph.
span	An integer that indicates whether this paragraph should span multiple columns when printing using a multi-column format. (*Not currently implemented.*)
float_how	An integer that indicates whether this object should float. Legal values are 0 (don't float), 1 (float if the object does not fit), 2 (always float). (*Not currently implemented.*)
float_where	An integer that indicates where this object should float. Legal values are 0 (float to edge), 1 (float to bottom), and 2 (float to top). (*Not currently implemented.*)

The attributes of a list format affect the format of objects embedded within the list. Lists can be used simply for grouping purposes (allowing additional editing functionality or attachment of semantic information) or they can be used for providing tagged enumerations or itemizations. The contents of a tag can be automatically generated based on an attribute of the list format or may be specified by the user. Attributes associated with a list format are:

leftmargin	Amount to indent any objects embedded in this list. Tags on a list are not indented by this margin. The actual left margin of a paragraph is the sum of the margin of the text formatting style and the margins of all the lists in which the paragraph is embedded.
rightmargin	Additional right margin for any elements in the list.
above	White space before the entire list (not each element in the list).

below	White space following the entire list.
breakpage	When printing, issue a page break before the first element in the list.
keep	When printing, attempt to keep the entire list on a single page.
label	How to generate the tags on the list items. The label field is a list of generator codes, each separated by a comma. Which code is used depends on the nesting of the list. The generator code is a combination of plain text and special generators that expands when generating the text. Legal generators are:

@1	1, 2, 3, 4...
@A	A, B, C, D...
@a	a, b, c, d...
@I	I, II, III, IV...
@i	i, ii, iii, iv...
@	You can edit the contents of the tag.

The default string defined for the `enumeration` type is `@1.,@a.,@i..` The default defined for the `itemization` type is `o,+,-.` Nested enumerations look like the following:

1. First point
    a. sub-point a
    b. sub-point b
        i. Further nesting
        ii. Some more
2. Second point
3. Third point

labelformat	The format used when displaying the tags.
textformat	The default format to use as the first item in the list. The editor allows you to group a number of existing items under a particular list format or to simply add a list format. Since a list format has to be "grouping" something, this field is used to specify what the initial element should be in this context.

The format table must be ordered so that the definition of a format precedes its use.

In addition to these attributes, a format can be *nonprimal.* A nonprimal format is a special local modification to some defined format. For example, you may have defined a `paragraph` format that indents the first line. However, you do not want one particular paragraph to begin with an indentation. You can change it by making a local modification to the format rather than defining a completely new format with a new name. A nonprimal format has several special characteristics. The name of a nonprimal format and the name of the format on which it is based are the same; no other format can be based on a nonprimal format (since inheritance is by name); and there must be at least one *primal* format of a given name.

2.4.8 Counter Tables

Counters are used for automatic numbering of sections, figures, lists, and so on. The counter table defines the numbering sequences that are used within a document. There may be a number of different sequences to support separate numbering of elements such as sections and figures. The format of a counter table is:

```
table counter
   ...sequence of counter styles...
endtable counter
```

where a counter style is defined with the form:

```
<counter name> <superior counter> <level>
level one style
level two style
.
.
.
<empty line>
```

The `counter name` is some sequence of alphanumeric characters not including spaces. The `superior counter` is optional and indicates another counter on which this one depends. Within the document, the numbering sequence for this counter is reinitialized when an instance of the `superior counter` at the given `level` occurs. This is used to support features such as figure numbering sequences that restart at every chapter. The counter table should be organized so that counters are defined *before* the counters they depend on.

Each counter style can define the style for multiple levels of this counter. This is used to support multi-level numbering for items such as sections, subsections, and so on. Each level can contain literal text, the control sequence for specifying the numbering style, and references to other counters. The numbering style is specified in the following form:

```
@1 Arabic numerals (1, 2, 3, ...)
@A Uppercase letters (A, B, C, ...)
@a Lowercase letters (a, b, c, ...)
@I Uppercase Roman numerals (I, II, III, ...)
@i Lowercase Roman numerals (i, ii, iii, ...)
```

Only one instance of the numbering style specifier can occur in a style level.

A reference to another counter is specified in the following form:

```
@counter(name:level:style:value)
```

Trailing fields can be omitted. The *name* field is the name of counter. The *level* field is the level of that counter. The *style* field is one of `incr`, `noincr`, or `init`. The *value* field is a numeric value to use for this instance of the counter if the style is `init`.

The following is an example counter table:

```
table counter
figure section 1
@counter(section:1:noincr)-@1

section
@1
@counter(section:1:noincr).@1
@counter(section:2:noincr).@1
@counter(section:3:noincr).@1
@counter(section:4:noincr).@1

appendix
@A

endtable counter
```

This table defines three counters. The `figure` counter contains a single style that includes a non-incrementing instance of the level one `section` counter (essentially retrieving the current value of that counter) and then specifies a literal hyphen followed by the control sequence (@1) for Arabic numerals. This generates numbers such as (1-1, 1-2, 1-3, 2-1, 2-2, 2-3, ...)

The `section` counter defines five levels. The level one counter is a numbering sequence with Arabic numerals, the level two counter includes the current value of the level one counter and another numbering sequence (1.1, 1.2), and so on. The `section` counter could be written more succinctly as:

```
section
@1
@<.@1
```

The @< sequence is shorthand for a non-incrementing instance of the current counter at level n-1. Since the default for a counter beyond the number defined is to use the last one defined, counter styles for levels three through five are not required.

Counters that begin with a pound sign (#) are automatically generated for use in numbering list tags.

2.4.9 Table of Contents Table

The `Table of Contents` (TOC) table is used to specify the contents and format of automatically generated tables of contents. The format of a TOC table is:

```
table ofcontents
...sequence of TOC definitions...
endtable ofcontents
```

where a TOC definition starts with the line:

```
StartTOC
```

and may include the following fields. Each field is on a separate line and consists of the field name, a space, and then the contents of the field up to the end of the line.

Title	The title of this TOC, such as "Table of Contents" or "Table of Figures".
TitleStyle	The formatting style to use for the title when generating the TOC file.

BodyStyle	The formatting style to use for the body of the TOC. A TOC definition may include multiple `BodyStyle` fields for specifying the style of different levels of the TOC.
File	The filename to use for the generated TOC. If no file is specified, the TOC filename is derived from the filename of the document.
ContentCounter	Specifies the counter used to generate the TOC. The value of this field is the name of a counter and the level to include in the table. There may be multiple instances of this field in a TOC definition.

The following is an example of a TOC table that defines a single table generated from two levels of section counters and the appendix counter.

```
table ofcontents
Title Table of Contents
ContentCounter section 2
ContentCounter appendix 1
File my_toc.slt
endtable ofcontents
```

2.5 Document Hierarchy

The last part of a BBN/Slate document file is the flattened object hierarchy. The flattened hierarchy is a sequence of elements. Each element can be either a single media object, a list containing a sequence of elements, or a reference to another file containing a sequence of elements. The overall sequence is terminated with the following token:

```
<enddocument>
```

Note that only the initial sequence (not a nested one) is terminated with the `<enddocument>` token.

2.5.1 Groups

A group is of the form:

```
<group formatname [formatindex]>
<list>
   ...element sequence...
<endlist>
```

The `formatname` is the name of some *list*-type formatting style. The `formatindex` is the index into the format table. If the index is present then the name is redundant (in fact the index may point to a nonprimal format, which is not indicated by the name). However, the protocol does allow the index to be omitted (useful if it is being generated by some other tool). The name also makes the structure more apparent when viewing the output file as plain text.

The special name NULL is used to indicate an anonymous grouping with no associated formatting style.

2.5.2 References

References are of the form:

```
<include filename>
```

The `filename` should name a file containing a legal BBN/Slate document. Tables in the included document are merged with the document it is being included in. The name is interpreted relative to the current directory.

The Document Editor can read documents of this form, but it cannot write files with indirect references and does not maintain the file boundaries internally. You can use this facility to bring together a number of related documents (*e.g.,* chapters of a manual) for viewing or printing, while editing them as separate documents in order to allow concurrent access by multiple writers.

2.5.3 Object Framework

Each object includes some generic information as well as the object-specific data. The output form of an object is:

```
<media mediatype>
<list>
<outline outlineflag>
<position xoff yoff width height>
<link link uid>*
   ...other object attributes...
<data> or <filename file>
[<bytecount nbytes>]
   ...object specific data...
<endlist>
```

The `position` and `data` attributes of an object are required; all others are optional. A description of each of these fields follows.

media The type of object the current object is. *`Mediatype`* is a string; current legal values are `header`, `text`, `graphics`, `image`, `spreadsheet`, and `speech`.

outline Indicates whether the object is displayed with a box around it. It is either 0 (no outline) or 1 (outline).

position The *`xoffset`* and *`yoffset`* are used when laying out a sequence of objects. (See Section 2.7 for a discussion of the algorithm used for laying out objects.) *`Width`* and *`height`* are self-explanatory. Units are expressed in the document resolution.

link	A link id. The semantics of linking is media-specific. The editor provides limited support — an object can access the objects it is linked to and can determine whether it is the first object among a number of linked objects to be read or written. This is useful for ensuring that the data that multiple objects use is only written out once. For example, when a document contains multiple views of the same spreadsheet the actual spreadsheet model would only be written out once, but the information describing which part of the sheet is visible could be written out for each view. Links are currently only used to support the connection between a spreadsheet and the charts it generates.
data	Marks the beginning of media-specific data. The data that follows may be either ASCII or binary format. The `<endlist>\n` marks the end of the data.
filename	The name of a file containing the media-specific data. This field would replace the `data` field of an object. The current editor supports this feature at the read-write level, but provides no mechanism to specify it within the user interface.

Other attributes may be specified but are not required. These include (but are not limited to):

justify	Indicates how the current object should be justified on the page (0 = left justification, 1 = center, and 2 = right).
breakpage	Indicates whether a page break should be issued before the current object (0 = no, 1 = yes, 2 = break to an odd page, 3 = break to an even page).
nobreakafter	Keeps the current object on the same page as the object that follows it. (This can be used to ensure that a figure stays on the same page as its figure label).
border_left	Amount of white space to reserve on the left of the current object.

border_right	Amount of white space to reserve on the right of the current object.
border_above	Amount of white space to reserve above the current object.
border_below	Amount of white space to reserve below the current object.
classify	An integer that indicates the classification level of the current object.
<xentry *id*...>	Indicates the start of an index entry. The full format is: `<xentry id flags page value>` where *`flags`* and *`page`* are integers and *`value`* is any text.
</xentry *id*>	The end of the index entry with the associated id. Index entries may be nested.

2.6 Representation of Specific Media Types

This section describes the format of the files used to represent each of the eight media types in BBN/Slate:

- Headers
- Text
- Graphics
- Images
- Rasterfiles
- Spreadsheets/Charts
- Speech
- Enclosures

2.6.1 Headers

Header fields are stored as name-value pairs. The format of the header data is:

```
font namefont valuefont
name1: value1
    .
    .
empty line
```

The first line specifies the font (as an index into the font delta table) used for displaying and printing the header-field names and values. Subsequent lines contain the actual names and values; names terminate in a colon, values terminate at a new line. An empty line concludes the data.

2.6.2 Text

The data associated with a text object may describe multiple sequential text objects. The general form is:

```
<protocol bbn>
<version 1>
<e formatname [formatindex]>
    ...text...
<e formatname [formatindex]>
    ...text...
</e>
```

Each element that begins with `<e` is a separate object in the normal sense of the word, but are all nested within the data of a single text object in the output format.

The *`formatname`* is the name of a *text*-style format.
The *`formatindex`* is an index into the format table. The

`formatname` is normally redundant and is only used when the `formatindex` is missing. This format is used to make the output more readable and to make it easier to generate a legal document from another program.

The actual text data is simply included verbatim. In order to prevent confusion between what is a format code and what is the actual text, a left angle bracket (<) or a back slash (\) at the beginning of a data line is delimited with a backslash.

Note that word-wrapped environments normally do *not* have embedded newlines stored in the data (although quoted newlines may have been inserted).

The beginning of a text element is marked by uppercase `<E` instead of lowercase `<e` if the text object has been automatically generated as the tag for a list item. Also, the end of the entire list of text elements may be marked by `</E>` instead of `</e>` when the document is the result of a *cut* operation and the cut did not include the complete paragraph. This produces a subtle semantic change when pasting within the editor.

In addtion to the information that specifies the formatting environment, there is information about font changes. A font change applies to a region of text. The start of the region is specified by a line:

```
<fn>
```

where *n* is an index into the fontdelta table. The end of the region is specified by the line:

```
</f>
```

Format changes cannot partially overlap (since an end marker is associated with the most recent font change), although they can be nested.

Note that both the font change information and the element-start notation start on a new line. The new line that preceeds the left angle bracket is not considered part of the data.

Additional format codes may be specified in the future to handle other formatting or semantic requirements. If you do not understand the format code, simply ignore it.

Other formatting codes include:

<link *id***>**	A link ID.
<oa *name value***>**	An object attribute.
<ro>	Indicates object is read-only.
<atomic>	Indicates the start of a region that is viewd by the editor as one entity. An atomic region must be deleted as a unit and you cannot insert into it. Atomic regions may not span multiple paragraphs. They are used for instances such as counter values.
</atomic>	The end of an atomic region.
<xentry *id***...>**	Indicates the start of an index entry. The full format is: `<xentry id flags page value>` where *flags* and *page* are integers and *value* is any text.
</xentry *id***>**	The end of the index entry with the associated *id*. Index entries may be nested.
<cnt ...>	An occurrence of, or a reference to a counter. The full format of this field is: `<cnt name type tag ntype style level init page>`

The `name` field is some previously defined counter. The `type` field is either 'o' for occurrence or 'r' for reference. The `tag` field is the symbolic tag used for referring to the current instance of the counter. Spaces in the tag as seen by the user are replaced with a tilde (~) in the output format. The special tag "----" indicates no tag. The `ntype` field is the numbering style used for the current instance of the counter (1, a, A, i, or I). It is actually just cached here since it could be derived from the counter style table. The `style` field is one of `incr`, `noincr`, or `init`. The `init` field is the value for the current counter if its style is `init`. The `page` field is the page where the current counter would occur in the printed output (-1 if it is not known). The `page` field is not guaranteed to be correct.

</cnt *n m*> Indicates the end of the automatically generated text for a counter. The integer *n* is the offset in the text buffer from the start of the counter text to the start of the actual counter number. The integer *m* is the offset from the start of the counter text to the character after the counter number. Nested `<cnt ...>` and `</cnt ...>` formatting codes count as one when determining this number. Counters may be (and usually are) nested.

<insert> An inserted object or document. The lines that follow this code make up the flattened object hierarchy for the inserted object or document. The document shares document tables with the document on which it has been inserted.

</insert> The end of an inserted object or document.

A note on performance and efficiency. Any system makes space-time tradeoffs based on assumptions about attributes of the data it is going to be working on. The multimedia editor assumes that paragraphs do not get "too big". If paragraphs become very large (perhaps because they were automatically generated by some other program) certain operations within the editor (such as formatting and redisplay) will be uncomfortably slow.

2.6.3 Graphics

A graphic object is stored in a "stylized" PostScript® format. The syntax used is legal PostScript but the operators in BBN/Slate provide a higher semantic level than afforded by raw PostScript. A stored graphics object is directly printable on a PostScript printer when prefixed by the BBN/Slate prolog and the associated texture and line tables.

Note that BBN/Slate does not accept and render arbitrary PostScript input. However, basing the output format on PostScript allows BBN/Slate development the ability to extend the range of input allowed in future releases without having to define the syntax. A benefit is that the stored format and the print format are the same.

Certain operators are used to record the editing state and do not actually cause any objects to be rendered:

gfx-protocol

```
string int gfx-protocol -
```

Specifies the protocol and version of the current object. Currently the protocol string is (Slate PostScript Graphics) and the version integer is 1.

arrows

```
boolean boolean arrows -
```

Indicates whether arrows should be placed on newly added lines. The first Boolean affects the first point of a line or line sequence; the second Boolean affects the last point.

rulers

```
int int int int int boolean rulers -
```

Specifies the attributes for rulers when editing. The arguments are, respectively, the major divisions, minor divisions, the increment (1 indicates inches, 0 indicates centimeters), the horizontal zero, the vertical zero and a Boolean value that indicates whether they are displayed or not.

grid `int boolean boolean grid -`

Specifies whether the grid is on. The first argument is the grid size (in units of 1/16 inches), the second is whether the grid is to be displayed, and the third is whether objects should be forced to align with the grid.

justification `int justification -`

Specifies the justification of newly inserted text. The integer may take on the value 0 (left justified), 1 (centered), or 2 (right justified).

scaling `scalefactor_n scalefactor_d resolution xoff yoff scaling -`

Specifies scaling and translation information that applies to each feature in the graphics oject. The arguments `scalefactor_n` and `scalefactor_d` are the numerator and denominator of a scaling factor. `Resolution` is the number of units per pixel. The `xoff` and `yoff` arguments are offsets from the top left corner of the object. Currently, `scalefactor_n`, `scalefactor_d`, and `resolution` always have the value 1.

The following operators cause objects to be rendered. Note that the resolution is assumed to be the resolution of the document and that the origin lies at the top-left.

line `linenum x1 y1 x2 y2 arrow1 arrow2 line -`

A line. The *`linenum`* is an index into a line table (see above) that specifies the width and texture of the line. A line is drawn from (*`x1,y1`*) to (*`x2,y2`*). *`Arrow1`* and *`arrow2`* are Boolean values that indicate whether arrows should be placed at the ends of the line.

lineseq

```
linenum x1 y1 ... xn-1 yn-1 xn yn npoints
    arrow1 arrown lineseq -
```

A line sequence. The *linenum* is an index into a line table. *Npoints* specifies the number of points (x,y pairs) in the line sequence.

freehand

```
linenum x1 y1 ... xn-1 yn-1 xn yn npoints
    arrow1 arrown freehand -
```

A freehand line sequence. The *linenum* is an index into a line table. *Npoints* specifies the number of points (x,y pairs) in the line sequence. While rendered the same as a lineseq, it has different editing characteristics.

freehandclosed

```
linenum x1 y1 ... xn-1 yn-1 xn yn npoints freehandclosed -
```

A closed freehand line sequence. The *linenum* is an index into a line table. *Npoints* specifies the number of points (x,y pairs) in the line sequence.

spline

```
linenum patnum x1 y1 ... xn-1 yn-1 xn yn
    npoints arrow1 arrown closed spline -
```

A spline. The points specified are the *knot* points of the spline. The system uses a cubic b-spline for rendering on the screen. The *closed* argument indicates whether the spline should be closed. If it is closed, it is filled with the pattern specified by *patnum*, an index into the texture table.

box

```
linenum patnum x1 y1 x2 y2 box -
```

A rectangle. The *linenum* is an index that specifies the line style to use for rendering the box and *patnum* is an index that specifies the pattern to use for filling the box.

roundbox

```
linenum patnum x1 y1 x2 y2 radius roundbox -
```

A rectangle with rounded corners. A radius of -1 is used to indicate that the proper radius should be derived from the size of the box.

polygon

```
linenum patnum x1 y1 ... xn-1 yn-1
    xn yn npoints polygon -
```

A polygon. The polygon may cross over itself.

circle

```
linenum patnum cx cy radius circle -
```

A circle of *radius* centered at *cx, cy*.

ellipse

```
linenum patnum cx cy dx dy ellipse -
```

An ellipse centered at *cx, cy* and with an *x* radius of *dx* and a *y* radius of *dy*.

chord

```
linenum cx cy radius angle1 angle2 chord -
```

An arc centered at *cx, cy* of *radius* from *angle1* to *angle2*. The angles are expressed in radians.

wedge

```
linenum patnum cx cy radius angle1 angle2 wedge -
```

A wedge centered at *cx, cy* of *radius* from *angle1* to *angle2*. The angles are expressed in radians. The wedge is filled with the pattern specified and the outline includes the lines from the arc endpoints to the center.

bitmap

```
x y datawidth dataheight width height bitmap data -
```

A bitmap image. The image origin (top-left) is moved to *x, y*. The *datawidth* and *dataheight* specify how many bits per row and how many rows. The *width* and *height* specify how large the image should appear. The *data* is a series of hex strings.

The following operators apply to text. The *fontnum* used in the operators is an index into the fontdelta table. The *y* coordinate specified in the operators below is the baseline of the text string.

text `x y fontnum justify string text -`

A string at the given position in the given font with the given justification. `Justify` is an integer, either 0 (left justified), 1 (centered), or 2 (right justified). The interpretation of the *x, y* position depends on the justification. The string is justified around it so for a left justified string, *x* marks the left edge of the string while for a right justified string *x* marks the right edge of the string.

left `fontnum x y string left -`

Puts a left justified string at *x, y*.

right `fontnum x y string right -`

Puts a right justified string at *x, y*.

center `fontnum x y string center -`

Puts a centered string at *x, y*.

repl `fontnum x y width string repl -`

Replicates the string enough times to fill the region from *x* to *x+width-1*.

filled `fontnum x y width string filled -`

Widens the width of the spaces in *string* in order to completely fill the region from *x* to *x+width-1*.

In addition to these operators, there are a few operators used for grouping objects. Groups may be nested.

group `procedure-body group -`

Causes the elements in the *procedure-body* to be rendered immediately. Within the editor, groups are recorded and their structure affects allowable editing operations.

groupdef

```
key procedure-body groupdef -
```

Causes the *procedure-body* to be bound to the given *key*.

groupexp

```
procedure x y scalefactor groupexp -
```

Causes the elements in *procedure* to be rendered. All the elements in the group are translated by the given *x* and *y* values and scaled by *scalefactor*. *Procedure* is presumed to actually be the value of some key bound with groupdef.

You can apply the lock operator after any feature to indicate it is locked.

2.6.4 Images

BBN/Slate currently uses the image format specified in RFC797. This is a binary representation, stored as 8-bit bytes.

The first 4 bytes of the file give the width of each line (in bits) and the next 4 bytes give the number of lines. These are stored in network order (most significant byte first). The ninth byte is the x increment and the tenth byte is the y increment. The x and y increment indicate how much room to leave between scan lines on the display: normally they are one.

Each line of the display is scanned from left to right and lines start from the top and move down. Each line ends on an octet boundary. If the width is not an divisable by 8, the rest of the last octet should be filled with zeros on the right.

2.6.5 Rasterfiles

The rasterfile media type is used to represent color images and images with depth greater than one. The format for color images in BBN/Slate documents is as follows. All integer quantities are stored in canonical network order (most significant bit first, most significant byte first) and converted to host order when the image is read into the editor.

Image Source (8 bits) A code indicating the source of the image. Legal values are:

0	Unknown.
1	Internet Request For Comments 797 format.
2	Sun Rasterfile
3	TIFF Revision 4.x
4	TIFF Revision 5.x
5	GIF image format
6	Macintosh PICT
7	MacII PICT2 format
8	MacPaint image, 72 DPI
9	Encapsulated PostScript
10	Scanned image

Image Type (8 bits) Type code indicating the type of the image. Legal values are:

1	Monochrome
2	RGB
3	Grayscale

Width (32 bits)

Height (32 bits)

Depth	(8 bits) Bits per pixel.
Data Bytes per line	(32 bits)
Horizontal Resolution	(32 bits) This is the horizontal resolution in pixels per inch. If 0, the screen resolution is assumed.
Vertical Resolution	(32 bits) This is the vertical resolution in pixels per inch. If 0, the screen resolution is assumed.
Aspect Ratio	(64 bits) The horizontal/vertical ratio, stored as an IEEE 64-bit floating point number in network byte order.
Flags	(32 bits) A set of flag bits. Values are:

0x1	External data
0x2	External colormap
0x4	Compressed image
0x8	CCITT 1D
0x10	CCITT Group 3
0x20	CCITT Group 4
0x40	TIFF RLE
0x80	TIFF Packwords
0x100	Packbits
0x200	GIF Lempel-Ziv-Welch
0x400	Sun rasterfile RLE
0x800	Microtek scanner

Colormap	(variable) The colormap is only present if the flag bits indicate that there is one. The format of colormaps is described below.
Data	(variable) Stored as 8-bit bytes, most significant bit first, most significant byte first. Color images are not compressed.

The format of a colormap is:

Colormap Type	(32 bits) This can currently take on the value 0 (indicating an RGB colormap) or 1 (indicating a monochrome colormap).
Colormap Size	(32 bits) Indicates the number of colormap entries in the table.
Red Count	(32 bits) The number of 8-bit values that follow to describe the red components of each color cell. This must be the same as the colormap size.
Red Values	(size * 8-bits)
Green Count	(32 bits)
Green Values	(size * 8-bits)
Blue Count	(32 bits)
Blue Values	(size * 8-bits)

2.6.6 Spreadsheets/Charts

The spreadsheet editor uses the Lotus 1-2-3^{TM} and Symphony^{TM} Worksheet File Format as described by documentation available from Lotus Development Corporation. BBN/Slate has some extensions to handle fonts and rulings and a few other elements, but these extensions follow the model described in the Lotus documentation.

For simple purposes (such as scanning past the data) you need to know the following:

- The worksheet format is a binary format.
- A worksheet is a series of records.
- Each record has a four byte header and a variable-length body.

- The first and second bytes of the record header is the record type code; least significant byte first.
- The third and fourth bytes of the record header encode the body length.
- The first record of the file is the *beginning-of-file* record, type code 0.
- The last record of the file is the *end-of-file* record; type code 1.

Although the editor reads and writes this binary format, it will also read documents that use a simple ASCII representation for tables with each row on a single line and columns separated by tabs. If this representation is used, no control is provided over other attributes like column width. This facility is provided for programs that want to automatically generate documents containing spreadsheets without worrying about the details of the binary representation.

2.6.7 Speech

Speech is encoded as a series of line-oriented records. There are eight required records and two optional records. Each record is encoded as either:

```
<keyword><newline>
```

or

```
<keyword><ascii space><data><newline>
```

Exactly one space must separate the keyword from the data. The records, in order, are:

```
encoding-type
recording parameters
layout width height caption-x caption-y icon-x icon-y
caption caption-text
style position-type justification-type fontindex
defaulticon or icon
icon-data    ( not present when icontype is default )
endicon      ( not present when icontype is default )
framecount frame-count framesize frame-size
frames
frame-data
endframe
```

Width, height, caption-x, caption-y, icon-x, icon-y, fontindex, frame-count and *frame-size* are all integers.

Encoding-type is currently one of "Watson Voice Server", "Natural MicroSystems VRX", "IBM RS/6000 ACPA V1.0", or "encoding lincoln_spp".

For encoding-types other than IBM RS/6000 ACPA V1.0 `recording parameters` is a line of the form:

```
Compression rate, Frame Size frame-size,
    Silence silence-suppressed
```

Currently, all parameters are zero.

For the IBM, recording parameters is a line of the form:

```
Rate rate, Frame Size frame-size, Channels channels
```

Rate is the number of samples per second per channel, *frame-size* is the number of bytes per frame and *channels* is the number of audio channels (one for mono, two for stereo).

Position-type is one of `above`, `below`, `left`, or `right`.

Justification-type is one of `left`, `right`, or `center`.

Icon-data is in the standard format as written by the image I/O routines.

Frame-data is the raw audio data. There should be *frame-count* * *frame-size* bytes of binary data.

2.6.8 Enclosures

Enclosures are represented in the BBN/Slate document output format by a special "start of enclosure" marker, followed by a series of enclosure components, terminated by a special "end of enclosure" marker.

The "start of enclosure" marker consists of the text `[Enclosure Start]` followed by a single new line character. The "end of enclosure" marker consists of the text `[Enclosure End]` followed by a single newline character. The case of the text is significant.

There are two types of enclosure components: simple components and compound components.

Simple Components

Simple components consist of one line of text of the form `name: value` followed by a new line. They are used to represent simple parts of an enclosure, like its type and caption.

There are a number of simple components used by BBN/Slate Release 1.1, which are listed below. In addition, if the document-reading code encounters a simple component whose name it does not recognize, it simply saves the line and writes it back out when the document is saved. This provides a measure of forward compatibility, allowing BBN/Slate to read and write enclosures that may be created by a future version of the editor.

Note that unrecognized lines may be moved to a different location within the enclosure object (in fact, they are all moved to the end when the enclosure is saved), so you should avoid order dependencies.

The following simple components are recognized by the BBN/Slate enclosure media type. Case is significant in component names.

Type Code

The type number of the current enclosure's type. *`Type code`* is an integer; the currently used type numbers are:

Data	0
Text	1
PostScript	2
Other	-1

Additional type codes may be added in the future, but "Other" will always remain -1. This component is mandatory.

Type Name

If the type code is -1 ("other"), this field contains the user-supplied name for the type. If this component is omitted, no user-supplied type will be assigned to this enclosure. *`Type name`* is a string.

Filename

The name of the file for the current enclosure, exactly as specified by the user. *`Filename`* is a string, and it is mandatory.

Description

The description for the current enclosure, exactly as specified by the user. If this component is omitted, no description will be displayed with the enclosure. *`Description`* is a string.

Flags

One or more words (separated by white space) that indicate which flags to apply to the current enclosure. The available flags are:

- `include-copy`
- `show-size`
- `copy-when-mailed`

If this field is omitted, a default set of flags will be applied. However, if the enclosure includes a copy of its data instead of a reference to a file, then the `include-copy` flag *must* be provided in this component.

Font Index	Index into the document's font tables of the font to use for the current enclosure's caption. If the value is 0, the default document font is used. If the component is omitted entirely, the default enclosure font is used. *`Font index`* is an integer.
Internal Border	Four integers, which are the number of pixels of white space to leave between the inside of the enclosure's outline and the text or icon for the enclosure. The four numbers of the number of pixels to leave at the top, left, bottom, and right of the enclosure, in that order. If this component is omitted, a default border is used.
External Border	Four integers, which are the number of pixels of white space to leave between the outside of the enclosure's outline and nearest other element of the document. The four numbers of the number of pixels to leave at the top, left, bottom, and right of the enclosure, in that order. If this component is omitted, a default border is used.
Icon Width	The width of the icon for the current enclosure (in pixels), if one is not using the entire contents of the 'Icon' compound component. If this component is omitted, the entire width of the 'Icon' component is used. Otherwise, the leftmost "icon width" columns are used. *`Icon width`* is an integer.
Icon Height	The height of the icon for this enclosure (in pixels), if one is not using the entire contents of the 'Icon' compound component. If this component is omitted, the entire height of the 'Icon' component is used. Otherwise, the topmost "icon height" rows are used. *`Icon height`* is an integer.

Data Size

The number of bytes of data to read from the 'Data' compound component. This component may be omitted if the enclosure contains a reference to a file instead of a copy of the data. *`Data size`* is an unsigned long integer.

Compound Components

Compound components are multi-line objects that begin with a line of the form:

```
name {
```

followed by a new line, and end with a line consisting of a single curly brace (}) character followed by a new line. They are used to represent complex parts of an enclosure, like its icon or its data.

The format of the information between the opening curly brace ({) and the closing curly brace (}) is defined by the component itself. There are a number of compound components used by BBN/Slate, which are listed below. As with simple components, if the document-reading code encounters a compound component whose name it does not recognize, it simply reads lines until it finds the closing }, saves the data, and writes it back out when the document is saved.

Note that unrecognized compound components may be moved to a different location within the enclosure object (in fact, they are all moved to the end when when the enclosure is saved), so order dependencies should be avoided. The data within one compound component will not be reordered.

The following compound components are recognized by the enclosure media type. Case is significant in component names.

```
Command {
    Name: <string>
    Template: <string>
    [optional other simple or compound components...]
```

The command component stores the value of a single enclosure command. An enclosure will generally contain multiple command components.

The name and command string (as supplied by the user) are stored in the `Name`: and `Template`: fields of a command. The names `Edit`, `Execute`, and `Print` are standard names that appear in the enclosure specification dialog. Other than that, they are the same as any other enclosure commands.

There is currently a limit of seven commands per enclosure, three of which must be `Edit`, `Execute`, and `Print`. The other four may have user-defined names. If the editor finds more than seven commands in an enclosure, it will save the additional commands as though they were unrecognized compound objects. No data will be lost, but commands after the first seven will not be executable.

For forward compatibility, the command component will also read and save any other simple or compound components that occur before the closing } character. For instance, a future version of enclosures could implement flags and access control lists and modify the command component to look like:

```
Command {
     Name: Compress
     Template: compress %s
     Flags: trusted modifies-data asynchronous
     Access Control List {
         fred wilma barney
     }
}
```

The BBN/Slate Release 1.1 Document Editor will correctly read, interpret, and save such command components.

Icon {[icon data]}

Used to store custom icons for an enclosure. The icon data is in Sun iconedit format, and must include the initial iconedit comment line that contains the width and height of the icon represented by the data. Note that the icon component interacts with the icon width and icon Height components described earlier. If this component is omitted, the default enclosure icon will be used.

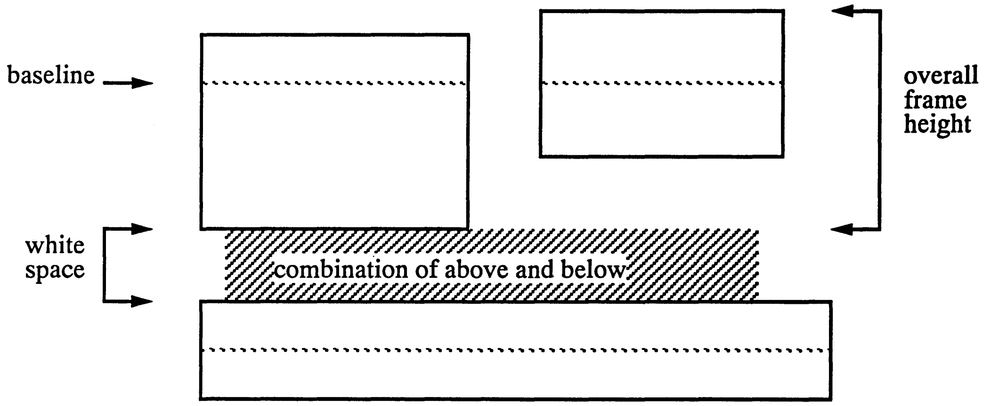
Data {[Data Size bytes of data]}

Used to store the enclosure's data, if the `include-copy` flag is set. This component contains data Size bytes of uninterpreted binary data. This component should be omitted if the enclosure contains a reference to a file, rather than a copy of its data.

2.7 Layout

A certain layout algorithm is assumed when a document is displayed on the screen or printed on paper. A document is a hierarchy of objects. Each object has a concrete size made up of nine parameters: width, height, x offset, y offset, baseline, left white border, right white border, top white border, and bottom white border. In addition, an internal node of the hierarchy can also have a top and bottom white border which serves as a lower bound for the top border of the top-most elements and the bottom border of the bottom-most elements in the list respectively.

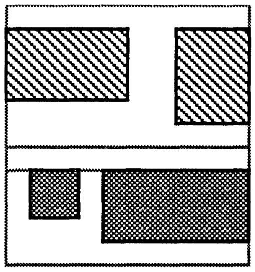
A sequence of objects which, given their x and y offsets, baseline and their width and height, do not overlap, are said to lie in the same *frame*.



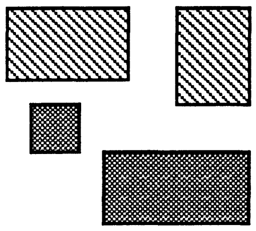
For example, the y offset of a paragraph is normally 0. The baseline is determined by the combination of fonts on the first line of the text block. The x offset is determined by the left margin. The width is determined by the left and right margins and the overall page width. The height is determined by the actual contents of the paragraph. The white space above and below are attributes of the formatting style.

Most non-textual objects have a width and height which initially defaults to some value and then can be explicitly adjusted by the user. The x offset and y offset are initially 0, but within the editor can be adjusted to allow arbitrary relative placement.

A frame has an overall width, height and white border. The next frame does not overlap the overall area at all, as shown in the following example:



rather than this next one:



The white space around two frames *overlap*, so to determine the white space between two frames the system take the maximum of the *below* of the top one and the *above* of the bottom one.

This layout algorithm is especially useful since it works forwards and backwards. It is not necessary to know the absolute location of an object in order to determine the layout of the objects around it.

3 MMDeliver

BBN/Slate delivers multimedia mail first by encoding it in a compressed text format, and then by using your normal UNIX workstation mail system to send it to the intended recipients. When you receive BBN/Slate messages that are encoded this way, they must be *decoded* before you can see the original multimedia content.

BBN/Slate includes a program named *MMDeliver* that performs this decoding automatically. Most BBN/Slate users want to have all of their incoming mail examined by MMDeliver so that multimedia messages are automatically decoded and delivered to the appropriate mailbox. MMDeliver can also convert text messages into multimedia messages, if you set it up to do so. In many cases, someone else has already arranged your mail processing for you. In other cases, you will have to do it yourself. Your system administrator (or whoever installed BBN/Slate) should be able to provide you with the details that apply at your site.

This chapter describes:

- how you can arrange to receive multimedia mail using MMDeliver with two common mail systems (Section 3.1)
- how you can tailor MMDeliver's mail processing features (Section 3.2)

MMDeliver is based on a very general mechanism for examining messages and taking action based on their contents, so it can do far more than simply unpacking encoded BBN/Slate messages. By writing custom mail processing rules, you can have MMDeliver automatically perform various tasks such as: convert text messages into multimedia messages, file messages in folders, and pass messages to other programs for more sophisticated processing.

Before you can receive BBN/Slate multimedia mail, you must arrange for all of your incoming mail to be examined by MMDeliver, so that encoded multimedia messages are detected and decoded.

3.1 Receiving Mail with MMDeliver

MMDeliver can be installed on a system in a variety of ways. Some of them result in automatic mail handling for all users. Others work by having the system's mail administrator add user names to a list of known BBN/Slate users, while still others require that each user take explicit steps to have his or her mail forwarded to MMDeliver.

You can contact your system administrator and ask about the procedures that apply to your machine. If the system administrator has already arranged mail forwarding for you, you can refer to Section 3.2 to learn how you can tailor MMDeliver's mail processing features.

If you must arrange for your own mail forwarding, you need to find out which type of mail system your machine uses. The two most common mail systems are Sendmail and MMDF. If you use Sendmail, proceed to Section 3.1.1; if you use MMDF, skip to Section 3.1.2. These sections describe how you can set up your own mail forwarding.

If your machine uses a mail system other than Sendmail or MMDF, the instructions in Sections 3.1.1 and 3.1.2 do not apply. Instead, consult your site's mail system administrator for help.

3.1.1 Mail Forwarding Using Sendmail

To have all of your mail go to your BBN/Slate mailbox, with automatic unpacking of BBN/Slate mail and mixed text/multimedia mail (like mailer error messages), add a line of the following form in a file named `.forward` in your home directory:

```
"|/usr/slate/bin.sun3/mmdeliver yourname -m"
```

To keep text and multimedia mail separate, enter this line instead:

```
"|/usr/slate/bin.sun3/mmdeliver yourname"
```

Replace */usr/slate/bin.sun3* in these examples with a path to the BBN/Slate binaries that run on your host.

To have some logging information recorded as your mail is processed, add the flag `-l` at the end of the line. To have a *lot* of logging information recorded, use `-L` instead.

By default, logging information is recorded using the syslog mechanism, but you can specify an alternate log file as an argument to the `-l` or `-L` flag. A simple file locking protocol is used to obtain exclusive access to the log file while MMDeliver is working on a message, so competing processes should not interfere with each other. Note that log files quickly grow in size if left untended.

There are many other options that you can use to tailor MMDeliver's behavior. These are described in detail in Section 3.2.

3.1.2 Mail Forwarding Using MMDF

To have all of your mail go to your BBN/Slate mailbox, with automatic unpacking of BBN/Slate mail and mixed text/multimedia mail (like mailer error messages), add a line of the following form in a file named `.maildelivery` in your home directory:

```
default - pipe A /usr/slate/bin.sun3/mmdeliver yourname -m -
```

To keep text and multimedia mail separate, enter this line instead:

```
default - pipe A /usr/slate/bin.sun3/mmdeliver yourname -M
```

Replace */usr/slate/bin.sun3* in these examples with a path to the BBN/Slate binaries that run on your host.

The example lines shown above assume that your site keeps its MMDF files in a directory named `/usr/mmdf`, and that MMDF's `submit` program resides in `/usr/mmdf/lib/submit`. If your site has installed MMDF in a directory other than `/usr/mmdf`, follow the `-M` flag with the name of the directory in which the `submit` program is located.

MMDF imposes a certain degree of security on mail processing; therefore it refuses to read your `.maildelivery` file unless the UNIX file protections prohibit group and other access to the file. The easiest way to ensure this is to give the following command at the UNIX prompt after initially creating the file:

```
% chmod 600 .maildelivery
```

To have some logging information recorded as your mail is processed, add the flag `-l` at the end of the line. Use `-L` instead if you want a lot of logging information.

By default, logging information is recorded using the syslog mechanism, but you can specify an alternate log file as an argument to the `-l` or `-L` flag. A simple file-locking protocol is used to obtain exclusive access to the log file while MMDeliver is working on a message, so competing processes should not usually interfere with each other. Note that log files grow quickly in size if left untended.

There are many other options that can be used to tailor MMDeliver's behavior. These are described in detail in the following section.

3.2 Customizing Mail Handling

You can use the MMDeliver program to automate much of your mail handling by using custom rule specifications. There are mail processing commands to deliver mail to text or multimedia mailboxes (Section 3.2.1), convert between text and BBN/Slate formats, file messages in folders automatically, discard mail you don't want to read, and other tasks. All of these options are described detail in the sections that follow.

3.2.1 How MMDeliver Processes Mail

Mail processing in MMDeliver is based on comparing the headers of each incoming message with a set of rules for what to do with different types of mail. Each user may specify his or her own set of rules. In addition, the site administrator may specify a set a rules that apply to all users. MMDeliver also has some built-in rules that handle the most common cases automatically.

Individual users' rules are generally kept in a file named `.mmdeliver` in each user's home directory, although you can use the `-f` flag to specify a different rule file. Sitewide rules are kept in a file named `/usr/slate/lib/mmdeliver.config`.

Every rule has the following information associated with it:

- A pattern describing the message headers that mark a message as being of interest to this rule.
- An input type describing the kinds of data to which this rule can be applied. Examples include text messages, and BBN/Slate multimedia documents.
- A command to invoke on the incoming message. The command reads one message and may produce another message, possibly changing the message type in the process.
- An output type describing the kind of message the command produces.

(Each rule also contains some additional pieces of information, which are described in detail later in this chapter. This overview describes only the items listed above.)

The first thing MMDeliver does when it detects an incoming message is to extract the message headers and save them for use in selecting rules. It then scans the rules looking for one whose message header pattern matches the headers for the current message, and whose input type matches the current message type, which is initially text.

If a matching rule is found, MMDeliver applies the rule's command. It then replaces the current message with the command's output and the current message type with the rule's output type, and continues scanning the rules looking for another one whose header pattern matches the saved headers, and whose input type matches the new message type.

Once the last matching rule is found and executed, MMDeliver examines its output type and determines what to do with the resulting data. If the final rule's output type is text, the data is delivered into the user's text mailbox. If the output type is multimedia, the data is delivered into the user's multimedia mailbox. If the output type is none, the data is discarded.

3.2.2 Command Line Options

MMDeliver accepts the following arguments on the command line. Most of them are optional, but the username *must* be provided.

username

The user to whom the mail is being delivered. This argument must always be supplied, and should be the *first* argument to MMDeliver.

-d *mail_directory*

The full path to your multimedia mailbox. You only provide this option if you want MMDeliver to put multimedia messages somewhere other than in your normal multimedia mailbox.

-D *folder_directory*

The full path of a directory in which your multimedia mail folders are kept. This directory will be used whenever messages are filed by the `MM_FILE` command described in Section 3.2.4.

You only provide this option if you want MMDeliver to file messages somewhere other than in your default multimedia folder directory.

-f *rulefile*

The name of a rule file to use in dispatching mail. The format of the rule file is described in detail in Section 3.2.3. Note that there are built-in MMDeliver rules that handle *all* of the usual cases. The rule file is only necessary if you want messages automatically to be filed in folders, discarded, forwarded to programs, and so on.

If no `-f` flag is supplied, MMDeliver will look for rules in a file named `.mmdeliver` in your home directory, and also in a system-wide rules file named `/usr/slate/lib/mmdeliver.config`.

Either or both of these files may be absent. Neither is required for simple mail unpacking and error message processing.

-i *inputfile*

Specifies a file for MMDeliver to process. The file may contain multiple messages, separated by the standard Sendmail message delimiter (the string `From` at the beginning of a line). The message delimiter pattern may be changed using the `-p` flag.

Note that processing multiple messages from one file *requires* that you use `-i` *`filename`*. If this flag is not supplied, MMDeliver processes its standard input, and assumes that there is only one message in the input stream.

-p *[pattern]*

The message delimiter pattern, specified as a regular expression in the style used by the UNIX `ed` program. Refer to the UNIX manual pages on `ed` for more information on regular expressions.

When processing multiple messages, lines in the input stream that match this pattern are presumed to separate messages. This is most useful when processing files that were not produced by the text mail system, or when using MMDeliver with mail programs other than Sendmail.

If the pattern is omitted, MMDeliver assumes there is only one message in the input stream, and does not try to find any message delimiter lines.

The default value is the `/bin/mail` separator string:

```
"^From "
```

unless MMDF mode has been selected with the `-m` flag, in which case the default is to assume one message in the input and no message delimiter.

-l *[logfile]*	Log information to a file as mail is processed. If a filename is supplied, it is used as the log file. Otherwise, the syslog() routine is used to log messages. Note that logfiles grow without bound if untended. On most systems, however, syslog entries are automatically removed after a week has passed. Simple file locking is implemented for logfiles, so multiple users can share a single log. To prevent a process that is hung or dead from monopolizing the logfile, locks are advisory only. A copy of MMDeliver that has been waiting for a logfile longer than about 10 minutes will simply ignore the lock, and deliver its mail anyway. Error diagnostics regarding the file-locking mechanism are logged with syslog. If your logfile appears to be corrupted, check the syslog file and see if it might be a locking problem. Logfile locking depends on any user being able to create and remove lockfiles. Therefore, shared logfiles should always reside in directories that are accessible to everyone (mode 777).
-L *[logfile]*	Log even more information than the `-l` flag. In particular, show the message headers of every message processed, and the rules that MMDeliver applies to it.
-m	Automatically convert text messages into multimedia messages, and deposit them in the multimedia mailbox.
-t *[textmail_delivery _agent]*	Do not merge text messages into the multimedia mailbox. Instead, deliver them to the text mailbox, using the textmail delivery agent supplied on the command line. This is the default behavior for MMDeliver. Therefore, the `-t` flag should only be required if you need to set the textmail delivery agent to something other than the default value.

The delivery agent is expected to read the message, including RFC-822 headers, from standard input.

The delivery agent specified on the command line may include the special sequences `%s` and `%r`. These will be replaced with the message sender and recipient names, respectively. The recipient name is the username supplied to MMDeliver. The sender name may be specified with the `-s` flag.

If no sender name is specified on the command line, MMDeliver assumes that the message delimiter pattern is a standard **From** line of the type used by `/bin/mail` (i.e., that it matches a string of the form `From` *`person date...`*), and uses the second word of each message's delimiter line as the sender for that message.

The default delivery agent is `/bin/mail -r %s -d %r`, unless MMDF compatibility mode has been selected with the `-M` flag, in which case it is `/usr/mmdf/lib/submit` with appropriate options.

One of these two is almost always correct.

-T *success_code*
Specify the exit status that the `textmail_delivery_agent` uses to indicate successful mail processing. Most UNIX tools will exit with status zero to indicate success. However, some mail systems are known to use non-zero exit codes instead. The `-T` flag tells MMDeliver what the textmail delivery agent's idea of success is so that MMDeliver can attempt to detect and log failures.

-s *sender_name*
Explicitly specify the sender name for messages delivered to the text mailbox. See the description of the `-t` flag, above, for more information.

-x *exit_status*

Exit with the given status if asked to process the same message twice. If this option is provided, MMDeliver adds a pair of extra headers (`X-MMDelivered-For` and `X-MMDelivered-By`) to every message that is delivered to a mailbox. If asked to process a message that already contains such a header, it exits immediately with the given status.

This option is provided for mail programs that do not provide a way to bypass per-user mail processing commands and deliver text directly into a mailbox (e.g., MMDF). Without this option, each time MMDeliver tried to generate a text message, it would be handed to another copy of MMDeliver for processing, and you would never be able to deliver anything.

-M *[mmdf_lib_dir]*

Selects MMDF compatibility mode for use in the MMDF `.maildelivery` file. When MMDF mode is selected, the following things occur:

- Input is assumed to come from standard input; the `-i` flag is ignored.
- Input is assumed to contain only one message with no delimiter line; the effect is as if the `-p` flag had been provided with no argument.
- The local text delivery agent is set to `/usr/mmdf/lib/submit`, with appropriate arguments. If MMDF is installed somewhere other than `/usr/mmdf`, you should provide a path to the MMDF `lib` directory after the `-M` flag. The name `submit` will automatically be appended to this path, along with appropriate arguments for use with MMDeliver.
- The text delivery agent's success status is set to octal 011 (decimal 9); this is the correct value when the MMDF Submit program is called with the default options built into MMDeliver. If you change the definition of the textmail delivery agent with the `-t` flag, you may have to use the `-T` flag to change the success status as well.

- The `-x` option (exit if asked to process the same message twice) is activated; the exit status is set to a magic number that informs MMDF that this message has already been processed, and should now be delivered.

-c Check rulefile specifications for errors.

If this flag is supplied, no mail is processed. Instead, the rule files that *would* be used to process mail are checked for syntactic and semantic correctness.

Of course, it is not possible to find all potential problems this way (some things cannot be determined until runtime, and depend on the message being processed), but you can at least detect gross mistakes in specifying rules without waiting for delivery of incoming mail to fail before you learn of the problem.

-help Print a usage message listing these options on standard error, then exit.

3.2.3 Rule Files

As explained earlier, MMDeliver can read a file of rules for processing mail, and dispatch your mail to mailboxes, folders, programs, or the trashcan, automatically. By default, it looks for a systemwide rule file in `/usr/slate/lib/mmdeliver.config`, and then for a private rule file named `.mmdeliver` in your home directory.

A rule file consists of a series of lines. Blank lines are permitted in the file, as are comment lines, and special lines that allow you to augment or override the options that were specified on the command line. Any other lines in the file specify mail processing rules, one rule per line.

Comments begin with a pound sign (#) as the first character on the line, and continue to the end of the line. Option lines begin with the string `options:`, and are described more fully below. Rule specification lines are also described below and in subsequent sections.

Specifying Options

Some system administrators choose to install MMDeliver as the standard mail delivery system for their sites, with a standard set of command line flags. Individual users, however, may wish to add to, or override, the options selected by the system administrator.

MMDeliver allows you to override the command line flags by including lines of the form:

```
options: command line options
```

in your private `.mmdeliver` file. It will read the file, looking for such lines and applying any options it finds, *before* trying to process any messages. You may include multiple `options:` lines in your `.mmdeliver` file, and they may appear at any point in the file, but MMDeliver will always read and follow all such lines before processing any mail.

The options in an `options:` line are specified exactly as they would be if they appeared on the command line. It is probably not advisable to include the `-f` option within your `.mmdeliver` file, because this causes MMDeliver to switch to a different rule file in midstream, and your original file is ignored.

Specifying Mail Processing Rules

Each rule specification consists of six fields, separated by colons. The fields are as follows:

priority

An integer between 0 and 99, inclusive. Rules with lower priority numbers are applied first. If two rules have equal priority, the one that appeared first in the file is applied first.

Built-in rules are applied at priorities between 5 and 10. User-specified rules should normally begin at priority 20, unless they want to specifically override built-in rules.

header specifications

MMDeliver uses a combination of the priority number and the messages headers to determine which rule(s) should be applied to each incoming message. For each rule, you may specify a set of header names and values that must be present (or absent) before the rule will be applied.

Names and values are specified as a comma-separated list of items. Each item has the form *name=value* if the header field must be present for the rule to apply, or *!name=value* if the header field must be absent.

The name and value are both regular expressions as defined by the UNIX `ed` program conventions, except that lowercase letters in the name and value match either uppercase or lowercase letters in the message. Uppercase letters in name and value match only uppercase letters in the message.

See the UNIX manual pages on `ed` for more information about regular expressions.

Examples:

The following specification matches any message in which the subject field contains the string `test message`:

```
subject=.*test message.*
```

This specification matches any message *not* from the system mail daemon:

```
!from=.*Mailer Daemon.*
```

This specification matches any message from root whose subject does *not* begin with the string `Undeliverable Mail`:

```
from=root,!subject=^Undeliverable Mail.*
```

As a convenience, the header specification field may contain the single character "-", which matches all messages.

input type

The type of data on which the command expects to operate. Legal values are `text`, `mm`, and `any`.

- `text` means this command will be applied only to text messages.
- `mm` means this command will be applied only to BBN/Slate multimedia messages.
- `any` means this command may be applied to any kind of message.

The initial type of all incoming messages is text. Note that a text message may have ASCII encodings for other content types embedded within it. Such embedded content is generally decoded by one of the MMDeliver rules, and the message type changed to match the type of the decoded data.

command

One of the built-in mail processing commands described below, or the name of some arbitrary program that will be invoked and fed the mail message on its standard input. Such external commands can produce new messages on their standard output, which are subject to further processing by MMDeliver.

See Section 3.2.4 for complete information on both internal and external mail handling capabilities.

output type

The type of data the command produces as its output. Legal values are `text`, `mm`, `discard`, and `any`.

- `text` means that the command produces another text message, suitable for further processing by rules that take `text` as their input type.
- `mm` implies that the command produces a BBN/Slate multimedia document, suitable for further processing by rules that take `mm` as their input type.
- `any` is used only with the `DELIVER` command described in Section 3.2.4.
- `none` means that this command produces no interesting output, and message processing should stop after the command is executed. An output type of `none` is normally used only with commands that explicitly deposit messages into folders or mailboxes.

input disposition

What to do with the input to this command after the command has been executed. Legal values are `save` and `discard`.

- `save` means that the input to this command should be saved in your mailbox (text or multimedia, depending on the input type and whether the `-m` flag was specified on the command line).
- `discard` means that the input to this command should be thrown away.

3.2.4 Mail Processing Commands

This section describes the seven built-in mail processing commands in MMDeliver. The use of a built-in command is indicated by placing one of the following keywords (possibly with an associated argument) in the `command` field of a rule.

DELIVER

Deliver the current text message to an appropriate mailbox. The message contents are analyzed to determine which mailbox is most appropriate for this message.

- If the message contains any encoded multimedia content, it is converted to a pure multimedia message and delivered to your multimedia mailbox. This includes mixed text and multimedia content within a single message, as might be generated by a mailer returning undeliverable encoded BBN/Slate mail, along with an indication of the errors encountered in attempting delivery.
- If the message body appears to contain only text, and the `-m` flag was not supplied on the command line or in your `.mmdeliver` file, then the message is delivered to your text mailbox.
- If the message body appears to contain only text, and the `-m` flag was supplied on the command line or in your `.mmdeliver` file, the text message is converted to a multimedia message and delivered to your multimedia mailbox.

The `DELIVER` keyword must be specified with input type `text` and output type `any`. At present, the `DELIVER` keyword does not actually produce output, aside from delivering the message to a mailbox. Executing a `DELIVER` rule always terminates processing of a single message.

DELIVER_TEXT

Deliver the current message to your text mailbox. The text mail delivery agent is invoked and supplied with the current message as its standard input.

Most mail delivery agents will expect the message to consist of a set of message headers conforming to Internet Request for Comments Number 822 (RFC822), followed by a blank line, followed by the message body. However, there is no enforcement of this policy within MMDeliver. It simply trusts you to choose appropriate mail delivery software and produce appropriately formatted files from any locally supplied message-processing tools.

The `DELIVER_TEXT` keyword must be specified with input type `text` and output type `text`. Like the `DELIVER` keyword, it does not produce any actual output, and executing it always terminates message processing.

MM_UNPACK

Unpack BBN/Slate multimedia mail that was encoded in the standard format by the editor or by the mmencode utility program. The result is a multimedia document suitable for processing by rules whose input type is `mm`.

This command implicitly assumes that it is being called in response to the detection of an **X-Content-Type:** header in the message. In particular, if such a header exists in the message, it is removed.

The `MM_UNPACK` keyword must be specified with input type `text` and output type `mm`. If it is the last filter processed, the output message will be delivered to your multimedia mailbox.

MM_FILE *folder*

Deliver the current (multimedia) message to a multimedia folder. The folder name must be separated from the `MM_FILE` keyword by exactly one space. If the name is preceded by a plus sign (+) character, it refers to a folder in your normal multimedia folder collection. Otherwise, it should be the full pathname of the directory in which the message will be stored.

The MM_FILE keyword must be specified with input type mm. The output type may be either mm or none. If mm is used, the message is delivered to the named folder, and rule lookup continues normally. If none is used, processing terminates after the message has been filed.

TEXT_TO_MM

Convert a text message to BBN/Slate multimedia format for further processing. This command adds an **X-Translated:** header to the message to record the fact that the message originated as simple text mail.

The TEXT_TO_MM keyword must be specified with input type text and output type mm. If it is the last filter processed, the output message will be delivered to your multimedia mailbox.

DISCARD

Discard the message immediately. This command may be specified with any of the valid input types, but the output type must be none. Obviously, executing a DISCARD rule terminates processing for the current message.

TRANSLATE *program*

Pass the message body to an arbitrary translator program, to deal with message formats not built in to MMDeliver. The program name must be separated from the TRANSLATE keyword by exactly one space.

The translator program is assumed to read from standard input and generate a BBN/Slate multimedia document, including a set of message headers, on standard output. The information passed in on standard input consists of a set of RFC-822 headers, followed by a blank line, followed by the message body.

The headers in the output document will be replaced by the original message headers (the same ones supplied to the translator on standard input). They are provided to the translator in case it wants to try to extract useful information from them to aid in the translation process.

Like MM_UNPACK, this command implicitly assumes that it is being called in response to an **X-Content-Type:** header in the original message. If such a header exists, it will be deleted. In any case, an **X-Translated:** header will be added to the output message to record the means by which it was created.

The TRANSLATE keyword can be specified with any valid input type, but the output type must be mm.

Any command field that does not match one of these built-in commands is assumed to be an external mail processor. In this case, MMDeliver tries to directly execute the named command, passing it the complete message (including headers) on its standard input.

External commands of this type can be specified with any valid input type. The output type must be text, mm, or none. An output type of any may not be used by an external mail processor.

The differences between this type of command and a TRANSLATE command are:

- External mail commands do not delete **X-Content-Type** header from the input.
- Nothing is assumed about the output format, and there is no attempt to insert the headers of the original message in the output, even if it is a multimedia document.

3.2.5 Default Rules

MMDeliver has the following rules built in:

```
5:x-content-type=X-BBN-Encoded-Multimedia;1.0:text:
    MM_UNPACK:mm:discard
10:-:text:TEXT_TO_MM:mm:discard
100:-:text:DELIVER:any:discard
```

Note that the TEXT_TO_MM rule is included only if the -m flag was supplied on the command line. User-supplied rules usually occur between the priority 10 TEXT_TO_MM rule and the priority 100 DELIVER rule.

These rules implement the default behavior of MMDeliver:

- At a very early stage (priority 5), BBN/Slate multimedia messages encoded by the Document Editor are unpacked.
- At a slightly later stage (priority 10), text messages are converted to multimedia messages, if requested via the -m flag.
- After the optional user-specified rules have been executed (priority 100), any text messages that have not been otherwise delivered are processed by the DELIVER command.

3.2.6 Examples of Rule Specifications

The rule-based handling of messages provided by MMDeliver enables you to manage a wide variety of message types automatically, but it can take some time to master the rule-specification syntax. These examples show how you might use MMDeliver to perform a variety of tasks with your incoming mail.

1. The following example simply converts all incoming text mail to multimedia form. This is similar to the built-in rule that is invoked when you use the -m option with MMDeliver:

   ```
   20:-:text:TEXT_TO_MM:mm:discard
   ```

2. This example automatically files multimedia messages that contain movie reviews in a folder called +movies:

   ```
   20:subject=.*movie review.*:mm:MM_FILE +movies:none:discard
   ```

3. This example files multimedia movie reviews in the +movies folder, but also delivers them to your multimedia mailbox so they appear as new mail:

   ```
   20:subject=.*movie review.*:mm:MM_FILE +movies:none:save
   ```

Note the use of the save input disposition to force a copy of the message into your multimedia mailbox.

4. Assume that MMDeliver is invoked without the -m flag, so that text and multimedia messages are treated separately. This example takes text messages containing movie reviews and translates them into multimedia messages in the first rule, then delivers them to the +movies folder in the second rule. Other text messages are delivered to your text mailbox.

```
20:subject=.*movie  review.*:text:TEXT_TO_MM:mm:discard
30:subject=.*movie  review.*:mm:MM_FILE  +movies:none:discard
```

   This example shows how you chain rules together so that the output of one rule becomes the input for the next.

5. Assume now that you are exchanging mail with users of another multimedia document system, like the Andrew Message System at Carnegie-Mellon University (which stamps its messages with an **X-Content-Type** header containing the string X-BE2;12).

   If you had a program to translate Andrew messages to BBN/Slate format, you could automatically accept messages from CMU and read them using BBN/Slate, with a rule like this:

```
20:x-content-type=X-BE2;12:text:
    TRANSLATE be2_to_slate:mm:discard
```

   You could file Andrew-format movie reviews in our +movies folder by using:

```
20:x-content-type=X-BE2;12:text:
    TRANSLATE be2_to_slate:mm:discard
25:subject=.*movie  review.*:mm:MM_FILE  +movies:none:discard
```

6. Consider how a cooperating group of BBN/Slate users could make use of custom message headers with MMDeliver.

   For instance, your group could agree on a standard mail header called X-Msg-Type to indicate what kind of information your messages contain. You could then use external mail processors to act on these messages according to their types.

For example, you might use:

```
20:x-msg-type=design  document:text:lpr:none:save
20:x-msg-type=design  document:mm:mmpr:none:discard
```

to print all design documents (using `lpr` to print text messages and `mmpr` for multimedia messages), or the following:

```
25:x-msg-type=progress  report:any:
     update_pert_chart:none:discard
```

to automatically update your project-wide PERT charts in response to a progress report from one of your group members, or the following;

```
30:x-msg-type=status  inquiry:any:
     send_status_report:none:discard
```

to automatically mail back a status report in response to an inquiry from one of the group members. (Of course, your externally supplied send_status_report program would want to be careful about whom it was willing to reply to!)

Or you might use:

```
20:from=frank,x-msg-type=lunch  invitation:any:
     DISCARD:none:discard
```

to throw away all lunch invitations from Frank without ever reading them (assuming you don't really like Frank very much).

4 MMail

MMail is a program you can use to read and manage multimedia documents from an ordinary character terminal when a workstation is not available. For example, you might use MMail to read your multimedia mail by dialing in to your workplace computer system from home.

MMail provides commands for reading, printing, filing, and deleting multimedia documents in a directory or folder. If you want to work with the contents of your multimedia mailbox, you can start MMail simply by typing `mmail` at the UNIX shell prompt.

If you want to work with the contents of some other folder or directory, you can supply a folder or directory name on the command line when you start MMail. For example, typing the following command would cause MMail to look at the contents of your `+action` folder instead of your mailbox.

```
% mmail +action
```

When you invoke MMail, it first looks in your mailbox, folder, or directory to see if you have any new messages. If you do have new messages, MMail displays a list of their message summaries like the one shown in the following figure:

```
   9: N   14 Feb 90 tcrowley  toolkit changes  (7K)
  10: N   14 Feb 90 milazzo   Re: colormaps   (9K)
  11: N   15 Feb 90 mlandau   Idea for text hierarchy (8K)
  12: N   15 Feb 90 mlandau   Issues in porting Slate(13K)
  13: N   15 Feb 90 tcrowley  Button changes  (14K)
> 14: N   15 Feb 90 tcrowley  New index capability (16 K)
mmail>
```

MMail Listing of New Messages

After listing your new messages, MMail displays its prompt and is ready for you to type a command. Section 4.3 lists all of the commands you can use with MMail. In addition to these commands, you can also type a question mark (?) or `help` at the MMail prompt to see a list of commands, and a brief description of each one.

4.1 Message Lists

Most MMail commands that display, file, or print documents accept an optional list of specified messages on which to operate. If you do not specify a list of messages when you issue a command, the command will operate on the *current message* by default.

There is one current message, and it is usually the last message displayed on your terminal screen. When MMail lists message summaries, the current message is marked with a right angle bracket (>). For example, in the previous figure, message number 14 is the current message.

When you enter commands in MMail, you can refer to messages by their numbers or by specifying a set of criteria to select a set of messages. A *message list* is *either* a list of message numbers separated by spaces or commas, *or* a set of message selection criteria. Message selection criteria are described in detail in Section 4.2.

You can specify a list of message numbers that includes any of the following:

.	refers to the current message
$	refers to the last message in the folder or directory
n	refers to message number n in the message summary list

n:m	refers to a range of messages. The range consists of message numbers *n* through *m*, inclusive. For example, the range 3:6 refers to messages 3, 4, 5, and 6

You can combine message numbers, message ranges, a dot, and a dollar sign ($) to select a set of messages by number. For example, the message list "1 2 6:9 $" includes messages 1, 2, 6, 7, 8, 9, and the last message in the folder or directory.

4.2 Message Selection Criteria

A set of message selection criteria may include any of the following items:

-new	New messages, which have not yet been included in a message summary listing by any of the BBN/Slate programs.
-unseen	Messages that have been included in a message summary listing, but have not yet been displayed by the Document Manager, MMail, or one of the other multimedia document tools.
-del	Messages that have been marked for deletion.
-replied	Messages to which you have replied by mail.
-old	Messages that have been included in a message summary listing.
-seen	Messages that have been displayed by the Document Manager, MMail, or one of the other multimedia document tools.
-undel	Messages that have not been marked for deletion.
-noreply	Messages that you have not replied to.

-b *date*	Messages received before the given date. Most U.S. date formats are understood, as are some informal formats like "last Tuesday". If the date contains multiple words, you must surround it with double quotation marks. A detailed description of date and time specifiers for `-b` are listed in Section 6.2.1.
-a *date*	Messages received after the given date. You can combine this with the `-b` option to select messages received between two dates.
-f *user*	Messages received from the given user.
-s *subject*	Messages containing the *subject* string anywhere in their subject headers.
-sel *name:str*	Messages containing a header field named *name* that contains the string *str* anywhere within its value. You can use this to select messages based on arbitrary mail headers.

You can issue a command using multiple selection criteria. Messages then must match *all* of the criteria to be selected as shown in the following example:

```
   mmail> del -f smith -s schedule -a "last thursday"
```

This command deletes all messages from `smith` that have the word `schedule` in their subjects, and that were received after last Thursday.

4.3 MMail Commands

This section describes all of the commands you can use with MMail. Many of the commands accept one or more arguments. When an argument is optional, it is shown in italics, surrounded by square brackets. Unless otherwise noted, any command that accepts an optional *message list* argument will operate on the current message by default.

When entering commands in MMail, you can type just the first few characters of any command, as long as what you type is unambiguous.

The following commands are available in MMail.

`<message number>`
Causes the message with the specified message number to display on the terminal screen. The messages is converted to a text-only form, then displayed using a pager. The pager is determined by looking at the PAGER environment variable, or by using the UNIX command `more` as the default pager.

`copy` `[`*`message list`*`] folder`
Copies messages to a folder without removing them from the current folder, directory, or mailbox.

`s` `[`*`message list`*`] folder`
`save` `[`*`message list`*`] folder`
`move` `[`*`message list`*`] folder`
Moves messages to a folder and removes them from the current folder, directory, or mailbox.

d [*message list*]
delete [*message list*]
Marks messages for deletion. Messages are removed from the folder, directory, or mailbox when you switch to another folder, exit MMail, or use the `expunge` command.

dp [*message list*]
dt [*message list*]
Deletes the selected messages and displays the next message on the terminal screen. The next message is the message immediately following the last one you deleted.

undelete [*message list*]
Unmarks messages that have been marked for deletion.

expunge [*message list*]
Removes any messages in the *message list* have been marked for deletion from the folder, directory, or mailbox. If you do not supply a message list, the default is to expunge all deleted messages.

list [*message list*]
headers [*message list*]
Lists summaries of the selected messages. If you do not supply a message list, the contents of the entire folder, directory, or mailbox is listed. If the list is longer than one screen, it is displayed using a pager. The pager is determined by looking at the PAGER environment variable, or by using the UNIX command `more` as the default pager.

set [*message list*] status
Sets the status of the selected messages. Status is a single character that is one of the following: N (New), U (Unseen), D

(Deleted), `R` (Replied To), and space (normal). To type a space, you must surround it with double quotation marks, as in `set 3:14 " "`.

`send` `[`*`message list`*`]`
Sends one or more multimedia messages that were previously prepared. The messages must already include headers indicating the recipients, as the `send` command does not prompt for them.

`type` `[`*`message list`*`]`
`view` `[`*`message list`*`]`
`p` `[`*`message list`*`]`
Displays the selected messages on the terminal screen. Messages are converted to a text-only form, then displayed using a pager. The pager is determined by looking at the PAGER environment variable, or by using the UNIX command `more` as the default pager.

`next`
`<Return>`
Displays the next message. The `next` command works the same as simply pressing **Return.** See the `type` command for more information about displaying messages.

`prev`
`-`
Displays the previous message. See the `type` command for more information about displaying messages.

`print` `[`*`message list`*`]`
Prints the selected messages out on a printer. The Document Editor is used to do the actual printing, so print settings are derived from the message contents and your editor initialization file.

`r` `[`*`message list`*`]`
`reply` `[`*`message list`*`]`
Replies to the sender of one or more messages. For each selected message, you are asked to confirm that you really want to reply, then placed in a text editor to type in your reply message. When you quit the editor, you are given the choice of re-editing, sending, or canceling your reply.

`R` `[`*`message list`*`]`
`Reply` `[`*`message list`*`]`
Replies to all recipients of one or more messages. This command is similar to the `reply` command, except that your response is sent to both the sender of the original message and all of its other recipients, rather than just to the sender alone.

`ri` `[`*`message list`*`]`
Replies to the sender, and includes a copy of the original message in your reply. This command is similar to the `reply` command, except that your response automatically begins with a copy of the message to which you are replying.

`Ri` `[`*`message list`*`]`
Replies to all recipients, and includes a copy of the original message. This command is similar the `Reply`" command, except that your response automatically begins with a copy of the message to which you are replying.

`forward` `[`*`message list`*`]`
Forwards the selected messages to one or more users. This command is somewhat like the `ri` command, except that you are asked to specify the recipients of your message explicitly.

mail username [*username* ...]
Sends mail to the specified users. As with the reply commands, you edit your message using a text editor, and then have the choice of sending, canceling, or changing it.

survey [*folder*]
folder [*folder*]
file [*folder*]
Confirms changes to the current mail folder, then switches to the named folder or directory. If you do not provide a folder or directory name, any of these commands lists all of your existing mail folders.

newmail
Confirms changes to the current mail folder, then switches to your multimedia mailbox.

help
?
Prints a list of MMail commands, and a summary of the syntax for specifying message lists.

quit
Confirms changes to the current mail folder and exits. If there are any background tasks in progress (*e.g.,* printing documents), MMail waits for them to complete before terminating.

5 MMScan

The BBN/Slate image editor allows you to create freehand drawings and add graphical features to them. However, you can also include copies of a drawing, photograph, letter, or other printed item in a BBN/Slate document.

MMScan is a program that uses an inexpensive document scanner attached to an IBM Personal Computer or compatible to scan printed material and generate image files that you can incorporate into BBN/Slate documents. MMScan *runs on the personal computer, not on your BBN/Slate workstation.*

Using MMScan, you can scan black-and-white drawings or halftone images in a variety of sizes and resolutions. In addition, MMScan allows you to adjust the scanner's operating parameters to obtain the best possible results with a wide variety of scanned materials.

The MMScan program is simple enough for the novice user, but powerful enough for the more experienced user to take advantage of the scanner's various features. Helpful prompts appear throughout the process of setting up the scanner and scanning an image, and it is virtually impossible to make a mistake that you cannot correct.

This chapter contains two sections:

- an image scanning tutorial (Section 5.1)
- advanced scanning techniques (Section 5.2)

The MMScan program is installed on an IBM Personal Computer or compatible connected to a Microtek model MS-300A image scanner. Your site administrator (or whoever installed MMScan) should have arranged for the personal computer to have access to the mass storage device used by your BBN/Slate workstation, and should have established a local `image library directory` that is accessible from your workstation. Refer to the *Installation Instructions* for information on installing MMScan.

Note: Before you attempt to scan an image for the first time, check with your site administrator or another knowledgeable user about the conventions established for use at your site.

5.1 Basic Image Scanning Tutorial

This tutorial describes the procedure for scanning an image using MMScan. It assumes MMScan has already been installed.

1. *Find something to scan.* It can be a black-and-white drawing, a color picture from a magazine, or a selection from today's newspaper comics. For this tutorial demonstration, you should probably use a black-and-white picture, and choose something about the same size as a standard 8 1/2 by 11 inch sheet of paper.
2. *Find the personal computer that is running the* MMScan *program.* If it is not actually scanning an image, it will either be prompting for a username or asking you to select scanner settings. If it is asking for scanner settings and you do not think there is anyone using it, press the **Escape** key and it will prompt for a username.

3. *Type your username in the highlighted box and press* **Enter**. Your username will normally be the same as your UNIX login name, although you are free to choose any username you like, or even to have more than one. Usernames are constrained by the same rules as MS-DOS filenames, but if you try to enter anything illegal, MMScan will help you figure out what is wrong.

   The highlighted editing box works somewhat like the BBN/Slate text editor; that is, you can move the cursor back and forth using either the BBN/Slate keyboard editing commands or the arrow keys on the PC's numeric keypad.

   The **Home** and **End** keys take you to the beginning or end of the field, respectively. You can delete characters with **^D** or the **Del** key on the keypad, delete from the cursor to the end of the field with **^K**, cancel the deletion with **^Y**, delete the entire field with **^W**, and so on.

   You can toggle between insert and overstrike modes by using the **Ins** key on the keypad. When you are in insert mode, the cursor is a solid block; when you are in overstrike mode, the cursor is an underscore. You can also press **Escape**, which gives you the option of terminating the MMScan program.

   If you do not type anything within about 30 seconds, or if you pause for more than 30 seconds while typing in your name, the program will erase what you have typed and reset itself. This is to prevent you from typing something and then walking away and forgetting about it before you have finished scanning an image.

   When you type your username, MMScan looks for your personal image library directory. Your image directory resides on your workstation as a subdirectory of the shared image library directory. For example, if your site's local image library directory is `/usr/local/lib/images` and your username is "joe", your private image directory would be `/usr/local/lib/images/joe`.

If MMScan does not find a private image directory for you, it asks if you want to create one. If you do want to create one, just type "**y**" and press the **Enter** key. If you do not want to create one (for instance, because you've misspelled your username), type "**n**" and press **Enter** and the cursor moves back to the username field, and you can re-enter your username.

4. *Type in a name for your image file.* Once you're satisfied with your username and MMScan has found your image directory, it asks you to type a name for the image you are about to create. Once again, the cursor is placed in an editing field. You can type an image name, or press **Escape** to change the username. If you pause for more than 30 seconds, MMScan erases your username and image name and reinitializes itself.

   The name you choose for your image file must be a valid DOS filename, and MMScan makes sure that it is. If the name is already in use in your image directory, you are given the option of overwriting the old image or changing the name. Just follow the instructions on the screen.

5. *Position the page on the scanner and begin scanning.* After you select your user and image name, you receive a short set of instructions and a list of scanner attributes that you can change. These attributes are discussed in detail in Section 5.2; for this tutorial just ignore them and accept the defaults.

   To start the scanner, position the page you want to scan on the scanner sheet feeder with the top down and the front facing away from you, and type **^Enter**. There may be a short pause (10 or 20 seconds) while the scanner warms up.

   If nothing happens within about 60 seconds, there may be something wrong with your scanner or with MMScan. If this happens, check to make sure the scanner and software are properly installed, then check with your site administrator for help.

6. *Read your image into a BBN/Slate document.* Assuming your page went through the scanner and came out the other end, you should now be able to read your image into a BBN/Slate document. Using BBN/Slate, create a new document and add an image name of the form `libdir/imagename`, where *`libdir`* is the name of your private image library directory and *`imagename`* is the name you chose as your image file name. If MMScan is working correctly, your image should appear on the screen.

5.2 Advanced Image Scanning Techniques

To attain the best results scanning a wide range of input documents, you need to know how to adjust the scanner settings. There are six different parameters you can control from MMScan. See the *Microtek MS-300A Intelligent Image Scanner Operation Manual* or the *EyeStar User's Guide* that came with your scanner for more complete information about scanner operations.

The six scanning parameters you can set are as follows:

Scanner Mode

The Scanner mode may be set to Graphics or Line Art. In Graphics mode, the scanner attempts to reproduce variations in shading of the original image by analyzing the image in small parcels and representing each section by a pattern containing the same proportion of dark pixels as the original image. The size of each parcel in pixels is set by the Grayscale parameter.

In Line Art mode, the scanner simply renders each pixel as black or white, depending on how dark the corresponding pixel is in the original image.

Line Art mode is best for images that consist of only two colors, even if those colors are not black and white. By using the Contrast and Brightness settings, you can often make the scanner render the darker color as black and the lighter as white, with a gray effect where the colors mix together.

Graphics mode is best for color images, or images that have many different shades of gray.

Resolution

The scanner is capable of scanning documents at a range of resolutions from 75 pixels per inch to 300 pixels per inch. The recommended setting is 75 pixels per inch (25% of full resolution) because that resolution most closely approximates the resolution of the Sun workstation display.

Scanning at 75 pixels per inch is also somewhat faster than scanning at 300 pixels per inch (about 15 seconds per page, compared to about 50 seconds per page at 300 pixels per inch).

Page Length

The Page Length setting allows you to tell the scanner that the page is longer or shorter than the default 11 inches. Page length may be from three to 14 inches, in 1/8 inch increments. However, if you set a page length that is shorter than the actual physical page, the scanner may generate error messages at the end of scanning.

Brightness

The Brightness setting allows you to adjust the scanner's default brightness level. Higher settings mean the entire image comes out uniformly lighter, while lower settings mean the entire image comes out uniformly darker.

Contrast

The Contrast setting allows you to change the scanner's sensitivity to contrast in the original image.

Higher settings mean that light areas appear lighter than usual, and the scanner is more sensitive to light shading within darker areas. Lower settings mean that the light areas appear darker, and the scanner is less sensitive to differences in shading within darker areas.

In Line Art mode, lower settings mean that a pixel must be darker than usual to be rendered in black, and higher settings mean the opposite.

Grayscale

The Grayscale parameter sets the size of the grid used by the scanner in analyzing an image (and consequently, the number of levels of gray that can be generated).

The values displayed are of the form:

```
<grid size> <number of pixels> <optional mask number>
```

Where a mask number is supplied, it means that the scanner can use any of several different arrangements of pixels to produce gray shades. Each arrangement is indicated by a different mask number.

Using larger grid sizes provides more levels of gray, but causes edge details to be lost. Using smaller grid sizes retains detail along sharp edges, but loses the distinction between similar colors or shades in the original image.

The Grayscale setting is used only in Graphics mode.

Once you enter your username and image file name in MMScan, it displays the current set of scanner attributes, with the Scanner mode highlighted for editing. Each of the six attributes has associated with it a list of possible values from which you can choose. The following keyboard commands allow you to set values, move from item to item, initiate scanning, and so on.

Changing the value of an item

space or **+**	Move forward to the next choice for this item.
backspace or **-**	Move backward to the previous choice for this item.
PgDn	Move forward by a larger increment (1/2 inch in the page length field).
PgUp	Move backward by a larger increment (1/2 inch in the page length field).
^PgDn	Move forward by an even larger increment (1 inch in the page length field).
^PgUp	Move backward by an even larger increment (1 inch in the page length field).
Home	Move to the first possible choice for this item.
End	Move to the last possible choice for this item.
a-z, 0-9	Jump to the next choice that begins with the selected letter or number; the search for a matching choice begins just after the current choice and wraps around to the beginning if necessary.

Moving from item to item

Arrow Keys	Confirm the value of the current item and highlight another item for editing. In most cases, the up and left keys move to the "previous" item, while down and right keys move to the "next" item.
Enter	Synonymous with down arrow.

F1	Move directly to Scanner mode.
F2	Move directly to Brightness.
F3	Move directly to Resolution.
F4	Move directly to Contrast.
F5	Move directly to Page Length.
F6	Move directly to Grayscale.

Miscellaneous other keys

^Enter	Confirm all settings and begin scanning.
Escape	Abort selection of scanner parameters and return to the username field.
^R	Reset all scanner parameters to their default values.

Once you have changed the settings to something you like, make sure the page you want to scan is in the scanner page guide and press **^Enter.** Your document will be scanned and the resulting image placed in your private image directory, where it will be available to include in a BBN/Slate document.

When MMScan is finished scanning, it returns to the beginning of the cycle and asks for a username. Your username is still displayed, and can be confirmed by pressing **Enter**. If you confirm your username in this way, your image file name is displayed for confirmation and editing, and your scanner parameter settings are preserved. This makes it convenient to scan multi-page documents where the individual files are named, for example, "page1", "page2", and so on. As always, if you do not type anything within 30 seconds, MMScan reinitializes itself for the next user.

5.3 Hints, Tips, and Cautions

The following are a few hints than can make your use of MMScan easier and more productive:

- If you expect to be scanning images often, you may want to set up a UNIX environment variable that points to your private image directory on the server.

  For instance, if you were to set the variable `i` to the directory `/usr/local/lib/images/`*`yourname`*`,` you would be able to include images in BBN/Slate documents by giving their names as `$i/filename`.
- This chapter assumes the availability of PC-NFS as the remote virtual disk service for the PC. It may be possible for the skilled PC user to install MMScan with some other remote disk service, as long as it allows creation of directories and files on the remote disk using the standard MS-DOS system calls. However, it is not recommended and BBN does not support any remote disk services other than PC-NFS.
- It is also possible to install MMScan without any remote disk service at all, using the PC's hard disk to store images and transfer them to the workstation manually.

  To operate MMScan without remote disk service, start the program by typing:

  ```
  mmscan <options> DIRECTORY
  ```

  where *`DIRECTORY`* is an existing MS-DOS directory name (e.g., `c:\images`).

Image files can then be found in `directory\username\`*`filename`*, and can be transferred to a Sun workstation using tftp, Kermit, xmodem, procomm, or any file transfer system that supports full 8-bit binary file transfer.

- Image files tend to be large. You should remember to clean up your image library directory periodically.

6 Multimedia Command-Line Tools

In addition to the Document Manager and the MMail program, BBN/Slate includes a set of tools for managing collections of documents from the UNIX command line. Although you normally use the Document Manager when you are at a workstation, the command-line tools are useful when a workstation is not available (for example, when you are logged in over a telephone or network connection), or when you wish to manipulate documents from within UNIX shell scripts.

You can invoke all of these tools as normal UNIX shell commands. Many of them do not require a window system, so you can also use them from a computer terminal.

This chapter describes how to use the BBN/Slate command-line document management tools. Section 6.1 presents some important concepts concerning files, documents, and the mail system from the perspective of the UNIX command-line interface. Section 4.2 discusses the individual UNIX shell-based tools beginning with the standard option flags used to select messages in all of these tools, and then listing the tools that are available and the services they provide.

Additional documentation is available on-line in the UNIX manual pages supplied with the BBN/Slate system. If your site has installed the manual pages, you should be able to type a UNIX command of the form `man` *`tool`* to see a description of any of document management tools.

A typical use of the document handling tools is to list the new messages in a folder with the mmls command, then use the other commands to display, file, print, or reply to them. The following is an example of the mmls command and its output when it is invoked to display the new mail messages in a user's mailbox.

```
% mmls -i -new
  58: N   13 Jan 87  forsdick  Re: Slate                   (11
  59: N   13 Jan 87  forsdick  message from U of M         (21
  60: N   15 Jan 87  mlandau    Internet Domain Addresses (2K)
  61: N   16 Jan 87  travers    New graphics commands      (13
```

Example of the MMls Command

The -i argument tells mmls to list the contents of your multimedia mailbox, or "inbox", and the -new argument tells it to list only new messages.

This example shows four document summaries displayed by the mmls command. The reference number, status, date, author, subject, and document size are listed from left to right within each summary. The mmls command is described in detail in Section 6.2.2.

6.1 Terms and Concepts

6.1.1 Document Summaries

Documents in the BBN/Slate system are stored as files in the UNIX file system; each document is stored in a separate file. Any UNIX command that operates on a file can be used to manipulate a BBN/Slate document, but the BBN/Slate document management tools provide a uniform and more coherent set of capabilities for handling multimedia documents.

BBN/Slate documents, although stored as ordinary files, have some additional information associated with them for use by the document management programs. The collection of information about a document is referred to as a *document summary.* A document summary consists of the following six fields:

Reference number

The reference number provides one way to refer to a BBN/Slate document. Another way is to use the name of the file that contains the document. Because of the way the BBN/Slate mail system and document handling tools generate file names automatically, it is usually more convenient to use the reference number.

BBN/Slate calculates reference numbers on a directory-by-directory basis; the numbers in each directory begin at 1 and end at the number of BBN/Slate documents in that directory. BBN/Slate generates reference numbers for new documents the first time you look at them with any of the document management tools.

A document's reference number may change when you remove other documents from its directory with the `mmmv` or `mmexp` commands, or when you remove the file in which a document is stored. After removing documents from a directory, it is usually a good idea to issue the `mmls` command to obtain accurate reference numbers for the remaining documents before using any of the other document management tools.

Document status

The document handling tools set the status of a document, but you may modify it using the `mmset` command described below. BBN/Slate stores a document's status as a set of single characters called *flags*:

N	(New) The document has never been displayed by `mmls`, `mmail`, or the Document Manager.
U	(Unseen) The document has never been displayed by `mmed`, `mmview`, `mmail`, or the Document Manager.

D	(Deleted) The document is marked for deletion, and will be removed from the file system the next time `mmexp` is run.
R	(Replied To) The document is a mail message that you have answered.

Date

If a document has been sent or received as an electronic mail message, its date is the date on which it was mailed. If the document has never been sent or received as a mail message, its date is the date on which the document was last modified.

Author

If a document contains a document header, the author is the person listed in the `From` header field. If a document does not contain a header, the author is the owner of the file in which is the document is stored. (Document headers are usually used in conjunction with the electronic mail facilities of BBN/Slate, although you may add them to any multimedia document. See Chapter 12 of the *Reference Manual* for a discussion of headers and how to use them.)

Subject

If a document contains a document header, the subject is specified by the `Subject` header field. If a document does not contain a header, the subject is the name of the file in which the document is stored.

Size

The size of a document is its size in bytes. It is generally displayed in kilobytes, by rounding the size to the nearest thousand bytes and displaying that number, followed by a "K". The length of a document does not necessarily indicate its size. Images and rasterfiles, for example, can consume a great many bytes, even though they may appear small when displayed on the screen.

The BBN/Slate document management programs maintain a record of the documents in each directory. Information about the documents in a directory is kept in a file named `.slate_docs` in that directory. BBN/Slate updates this file whenever the contents of the directory changes.

If you remove the `.slate_docs` file from a directory, BBN/Slate will create a new one next time it is needed. However, you will lose document status information including which documents were marked for deletion. When the new `.slate_docs` file is created, all of the documents in the directory will revert to the `N` status.

6.1.2 Messages and Mailboxes

One of the features that makes BBN/Slate unique among multimedia document systems is its ability to handle documents as electronic mail. You can create documents and then mail them to other users of BBN/Slate. They may be mailed to other users on the same computer system or to users on a different computer system that is reachable over some communication network.

When you send a message to a BBN/Slate user as electronic mail, the mail system deposits it in a special directory known as the user's *mailbox.* Each mail message is a complete BBN/Slate document, so the mail system places each one in a separate file in the mailbox directory.

By convention, a user's mailbox is the specially named folder `+Mailbox`. To manage mailboxes more conveniently, there is a special flag (the `-i` flag, discussed in detail in Section 6.2.1) that you may pass to each of the document management programs to specify "use the documents in my BBN/Slate mailbox".

6.1.3 Folders

Because BBN/Slate documents are stored in ordinary files, you can organize collections of related documents simply by keeping them in the same directory. The BBN/Slate document management tools provide additional support for organizing documents through the use of *folders.*

A folder is a directory in which documents are stored. The two differences between folders and other directories are:

- Folders have a special naming syntax.

  Folder specifications always begin with a plus sign (+). The name `+wishlist`, for example, refers to the folder whose name is "wishlist", and not to a file or directory named `+wishlist`. Folder names themselves must adhere to the same rules as UNIX filenames.
- Folders are kept in a well-known location.

  Your folders are kept in a subdirectory named `Slate`, which is located in your home directory. When one of the document management tools sees a name such as `+wishlist`, it interprets the name as a folder specification and substitutes the directory `~/Slate/wishlist` (where the tilde character (~) refers to your home directory).

Using folders permits you to organize documents simply by specifying "move this document to the wishlist folder." You need not be concerned about where that folder is really located, nor remember and type long pathnames when referring to documents or collections of documents.

6.1.4 Folder Management

Folders are UNIX directories, and as such they may be manipulated with the usual UNIX commands that operate on directories (`mkdir`, `rmdir`, `mv`, `rm -r`, and so on) In addition, you can create new folders with the command-line document management tools. Most folder management issues are encompassed in the following five tasks:

Creating Folders

The document management programs will create a new folder automatically the first time you specify one that doesn't already exist. Therefore when you move or copy a document from one folder to another, BBN/Slate creates the destination folder if it does not already exist.

For the shell-based tools, simply specify a new folder as you would any existing folder and it will be created.

You may also explicitly create a folder by using the UNIX `mkdir` command. For example, to create a folder named `+action`, you could issue the following UNIX command:

```
% mkdir $HOME/Slate/action
```

Removing Folders

You can remove folders using the UNIX `rmdir` or `rm -r` commands. For example, to remove the folder `+action`, you use the following command:

```
% rm -r $HOME/Slate/action
```

Renaming Folders

You can rename folders using the UNIX `mv` command. For example, to change the name of the `+action` folder to `+urgent`, use the following command:

```
% mv $HOME/Slate/action $HOME/Slate/urgent
```

Providing Access to Documents in Other Directories

You can store BBN/Slate documents anywhere within the UNIX file system, not just in the BBN/Slate folder collection. It is often convenient to make documents stored in other directories *appear* to be part of your normal folder hierarchy. You can do this using UNIX *symbolic links* to create an entry in your `Slate` directory that actually refers to some other directory.

For example, if you had some documents (and other files as well) in a directory named `/usr/project/docs`, you could make these documents appear to be in a folder named `+project` by using the following UNIX command:

```
% ln -s /usr/project/docs $HOME/Slate/project
```

6.2 Using the Document Management Tools

6.2.1 Using Option Flags to Select Documents

The document management tools include commands for summarizing, editing, filing, and printing BBN/Slate multimedia documents. In general, each tool performs only one task.

All of the document management commands accept a set of standard option flags that specify the directory (or folder) to use, the documents upon which to operate, and the order in which to operate upon them. You can specify documents by one or more of the following criteria:

- file name
- reference number
- document status
- contents of document header fields
- date received or modified

Note these are the same items that are in a BBN/Slate document summary. The general syntax for invoking any BBN/Slate document management command is:

```
command [directory specifier] [message selectors]
        [command-specific information]
```

The *`directory specifier`* indicates which folder or directory is to be searched for documents.

The *`message selectors`* specify the documents on which the command should operate. Only documents that match the criteria listed in the message selectors are affected by the command; other documents in the same directory or folder remain untouched.

The *`command-specific information`* is detailed in Section 6.2.2, under the command to which it applies.

The directory specifier and the message selectors are both optional. If you do not supply a directory specifier, the document management tools described in this chapter will use the current directory. If you do not supply any message selectors, the commands will operate on all of the documents in the directory.

Directory Specifiers

The optional directory specifier may be one of the following:

-d *directory*	Operate on documents in the named directory. The directory may be a UNIX pathname or a folder name as described in Section 6.1.3.
-i	Operate on documents in your BBN/Slate multimedia mailbox. The `-i` directory specifier implies the `-n` message selector; *i.e.*, messages in your BBN/Slate mailbox are *always* specified by reference number, and *never* by filename.

If you do not include a directory specifier in a command, the command operates on documents in the current working directory. You cannot specify both the `-d` and `-i` flags in the same command.

Message Selectors

A complete list of message selectors is given below. The message selectors you specify may include any or all of those listed, with the following exceptions:

- Reference numbers and filenames may not be mixed in the same command.
- The `mmnew` and `mmexp` commands do not use message selectors.
- The `mmcp`, `mmdel`, `mmmv`, `mmset`, or `mmundel` commands do not use filenames. To move, copy, or delete a document by filename, use the UNIX file commands to do so.

When you specify more than one message selector, BBN/Slate selects only documents that match *all* of your specified criteria.

-n *reference numbers*

Use documents with the specified reference numbers.

Specify reference numbers as a list of numbers separated by spaces or commas, or as a range of the form `m:n`. A range is all numbers from `m` to `n`, inclusive. You can use the dollar sign character ($) in a list or range to specify "the highest numbered document in the directory."

You may mix lists and ranges. For example, the reference numbers `1,3,4:7,11,15:$` refer to document numbers 1, 3, 4, 5, 6, 7, 11, and 15 through the last available document.

If you use the `-i` directory specifier, you can omit the `-n` flag. When you use `-i`, BBN/Slate automatically interprets all document references as reference numbers.

filenames	Use documents in the named files. You can specify one or more filenames separated by spaces. If you use the `-i` directory specifier, you may not specify filenames. If you use filenames in a command, BBN/Slate ignores the `-d` directory specifier and looks for the named files relative to the current working directory.
-new	Use documents for which summaries have never been displayed, or whose status you have explicitly reset to `New`.
-old	Use documents for which summaries have previously been displayed.
-unseen	Use documents that have not been displayed in the multimedia Document Editor or with MMail, documents that have changed since they were last displayed, or documents whose status you have explicitly set to `Unseen`.
-seen	Use documents that have not changed since they were last displayed.
-del	Use documents that have been marked as deleted.
-undel	Use documents that have not been marked as deleted.
-replied	Use documents that have been replied to using the mail system, or whose status you have explicitly set to `Replied To`.
-noreply	Use documents that have not been replied to using the mail system.
-b *date*	Use documents received or modified before the given *date*. The date may be specified in any of several forms, including the following components:

Time	Specify a time in terms of hours, minutes, seconds, meridian (am or pm), and time zone. Hours are required, all other fields are optional. Examples include:

```
22:14
9:15:30
9:15 am GMT
```

Date	Specify a date in any of the following forms:

```
5/29/87
May 29, 1987
29 May 1987
29 May 87
```

You may spell out months or use the customary 3-letter abbreviations.

Day	You can spell out a day (e.g., `Monday`) or use the customary abbreviations. You can also use the words `yesterday`, `today`, `tomorrow`, and `now`.
Interval	Intervals include `second`, `minute`, `hour`, `day`, `month`, and `year`. You can use both singular and plural forms.
Relative Time	You can precede day names and intervals by a positive or negative number or the words `this`, *`next`*, or `last` to indicate which instance of the day or interval is meant. You can also use the word `ago`.

All of the following examples are valid date specifications:

```
5/29
May 29,1987
15 June 1976
9 am
11 pm yesterday
last Monday
10:15 am PST last Friday
2 weeks ago
next month
last year
```

If a date specification contains spaces, you must surround it with double quotation marks.

-a *date*

Use documents received or modified after the given *date*. Refer to the -b option for a description of what constitutes a date. If you specify both a "before date" and "after date" BBN/Slate selects documents as follows:

- If the before date is later than the after date, then BBN/Slate selects documents received between the two dates. For example, to select documents received between January 1 and June 15, use:

  ```
  -a 'January 1' -b 'June 15'
  ```

  This combination means "use documents that were received after January 1 *and* before June 15."

- If the before date is earlier than the after date, then BBN/Slate selects documents received before the before date *and* documents received after the after date. For example, to select documents received more than one week ago and documents received since yesterday, use:

  ```
  -a yesterday -b 'last week'
  ```

  This combination means "use documents that were received after yesterday *or* before last week."

-sel *name:value*

Use documents that contain the specified value in the document header field named *name.* The standard header field names are listed in the *Reference Manual*; you can use both the standard header field names and user defined names to select a set of documents.

Separate the header field name and value by a colon; separate multiple name/value pairs by commas or spaces. Header field name comparisons are case insensitive; header field value comparisons are performed using ed-style regular expression matching. (Refer to the UNIX manual page on the `ed` program for a description of `ed` program for a description of ed-style regular expressions.)

Where you specify more than one name/value pair BBN/Slate selects, only documents that match all of the criteria. For example, to select all documents from a user named `Travers` on the subject of `graphics`, use:

```
   -sel from:travers -sel subject:graphics
```

-sort *name*

This option does not select documents. Instead, it controls the order in which BBN/Slate displays and processes the selected documents. Specifying a sort option sorts the selected documents alphabetically based upon the contents of the name document header field.

For example, to sort documents based on the author's name, so that messages from the same individual are grouped together, use:

```
   -sort from
```

To sort documents based on the subject use:

```
   -sort subject
```

Sorting by subject has one special property. When a user replies to a mail message in BBN/Slate, the mail system adds the prefix `Re:` to the subject field of the reply. Sorting documents by subject explicitly ignores this initial `Re:` prefix, so that related mail messages will be grouped together.

-rev This option does not select documents. Instead, it reverses the order in which the selected documents will be displayed and processed.

The following examples are all legal commands using the specifiers in a variety of combinations:

```
mmls -d +slate-ref -sel subject:App -rev
mmls -i -new
mmls -i -b today -rev
mmdel -d +actions -b 1/1/88
mmcat example
mmcat example.slt
```

6.2.2 Document Management Functions

This section describes the document management functions available in BBN/Slate. Some of the commands invoke the Document Editor to display, edit, or compose replies to documents. You must run these commands from within your window system for the editor to be available. If you attempt to run a command that requires the editor from outside the window system, it will display an error message and terminate immediately.

Some of these commands also accept or require arguments in addition to those described in Section 6.2.1. Each command description lists whether you must run the command from within the window system and whether it accepts any command-specific options or arguments. Additional information about these commands is available from the UNIX on-line manual pages distributed with BBN/Slate.

mmcat	Creates a text-only version of a multimedia document. This command formats text for a page width of 79 characters or the value of the variable `text-plain-linewidth` using a fixed-width font. BBN/Slate attempts to preserve and represent lists, itemizations, enumerations, etc. It represents spreadsheets as columns of data aligned with spaces or tabs. It cannot convert images, graphics, and voice into text; it indicates these objects by a message in the text.

Requires window system:	no
Other arguments:	none

mmcp

Copies the selected documents to a file, folder, or directory. This command requires one command-specific argument, the destination into which documents should be copied. The destination must appear as the last argument on the command line. For example, to copy messages about bugs from a user's mailbox to a folder named `bug-reports`, type:

```
% mmcp -i -sel subject:bug +bug-reports
```

If only one document is selected, the destination may be a file, folder, or directory name. If more than one document is selected, the destination *must* be a folder or directory name. If the destination is a folder name, BBN/Slate creates the folder if it does not already exist.

Requires window system:	no
Other arguments:	a destination file or directory

mmdel

Marks the selected documents as being deleted. You may undelete them with `mmundel` so long as you have not physically removed the files containing them by using `mmexp` or the UNIX `rm` command.

The `mmdel` command normally requires confirmation before deleting unseen messages. Use the command-specific option `-bk` ("run in background") to override this behavior.

Deleted documents are not actually removed from the file system until the expunge command, `mmexp`, is run. Use the command-specific option `-exp` to force an immediate expunge.

Requires window system:	no
Other arguments:	`-bk` and `-exp` options

mmed

Invokes the multimedia editor to edit the selected documents. If multiple documents are selected, the first one appears in the editor; you may move back and forth through the documents with the editor's **Editor-File-Read-Next File** and **-Previous File** commands. If no documents are selected, the directory specifier is ignored and the editor is invoked to create a new document.

Requires window system:	yes
Other arguments:	none

mmexp

Removes (expunges) all documents that have been marked as deleted for the specified directory. Once expunged, deleted documents cannot be retrieved. Any document selector options are ignored by `mmexp` -- *all* deleted documents in the directory or folder are removed.

Requires window system:	no
Other arguments:	none

mmfwd

Invokes the multimedia editor to forward each of the selected documents to one or more additional recipients. When BBN/Slate reads each document into the editor, it automatically constructs a message template for a new message. The template contains document headers and a copy of the document being forwarded. Send forwarded messages by adding to the body of the message, if desired; by editing the header fields to set the recipient list and subject; and by invoking the editor's **Send** command.

If multiple documents are selected, the first one appears in the editor. You may forward each one in turn by using the **Editor-File-Read-Next File** command.

Requires window system:	yes
Other arguments:	none

mmls

Displays document summaries for each of the selected documents. Document summaries are described in Section 6.1.1. (See page 4-1 for an example of output from `mmls`.)

Although a document may have multiple status flags set at one time, `mmls` prints only one character for the document status. The rules BBN/Slate uses to determine that status displayed are:

- if the document is marked as deleted, the status `D` is given
- if the document is marked as new, the status `N` is given
- if the document is marked as unseen, the status `U` is given
- if the document has been replied to using the mail system, the status `R` is given
- if a user-defined status character has been assigned with the `mmset` command, that status character is given
- if none of these rules apply, no status is given

The `mmls` command also establishes reference numbers for new documents that appear in a directory or folder. If the contents of a directory or folder has changed recently, it is usually good practice to run `mmls` and obtain an up-to-date set of reference numbers before using any of the other document management programs.

Requires window system:	no
Other arguments:	none

mmmv

Moves the selected documents to a file, folder, or directory. This command requires one command-specific argument, the destination into which documents should be copied. The destination must appear as the last argument on the command line. For example, to move messages 1 through 6 from your multimedia mailbox to the directory `/usr/docs`, use:

```
% mmv -i 1:6 /usr/docs
```

If only one document is selected, the destination may be a file, folder, or directory name. If more than one document is selected, the destination *must* be a folder or directory name. If the destination is a folder name, BBN/Slate creates the folder if it does not already exist.

Requires window system:	no
Other arguments:	a destination file or folder

mmnew

Determines if the specified directory or folder contains any new documents. If there are new documents in the directory the message "You have new Slate multimedia mail" appears on your terminal. The document selector options are ignored by `mmnew`.

The command `mmnew -i` is typically included in a user's shell initialization file (`.login` for csh user, `.profile` for sh users) to check for new mail when the user logs in.

Requires window system:	no
Other arguments:	none

mmpr

Prints the selected documents on a PostScript-compatible printer, using the default printing parameters (page size, margins, headers and footers, etc.) that were established when the document was created.

To change the printing parameters, edit the document (using the **show** command from the Document Manager or the `mmed` command), and use the **Editor-Print/Preview** command to set the desired values.

Invoking `mmpr` causes BBN/Slate to generate a PostScript representation of each document and send it to a program called a *print spooler*. The print spooler should be the name of the program used to print PostScript files on the local machine.

The default print spooler was established when your system administrator installed BBN/Slate, but you can edit your `.slate_editor.init` file to change it. You can also use the environment variable `SLATE_PRINT` to override both the default value and the value in your editor configuration file, in case you need to use a special print spooler other than the one you normally use.

You can set this environment variable to the name of a program that can read PostScript code on its standard input and send that PostScript on to a printer. Setting the print spooler to the UNIX `cat` program causes the generated PostScript to appear on the standard output.

Requires window system:	no
Other arguments:	none

mmrply

Invokes the multimedia editor to construct and mail a reply to the author of the first selected document. When BBN/Slate reads each document into the editor, it automatically constructs a template for the reply. The template contains document headers with an appropriate **To** address and subject. You send a reply by entering the body of the reply into the template, editing the headers if necessary, and invoking the **Editor-Mail-Send-Send Message** command.

If multiple documents are selected, only the first one appears in the editor.

Requires window system:	yes
Other arguments:	none

mmset

Sets the status of the selected documents, *overwriting any status you or the document management system previously set.* You may set the status to any single-character value, but the values **N**, **U**, **D**, and R have special meaning to the document management tools, as described in Section 6.1.1.

Requires window system:	no
Other arguments:	Requires a single character status value as the last argument on the command line. Selected documents have their status set to that character. To clear the status, use empty double quotation marks as the last argument (" ").

mmsnd

Mails the selected documents to the recipients you have specified in their document headers.

Requires window system:	no
Other arguments:	none

mmundel

Undeletes the selected documents, if you have already marked them for deletion.

Requires window system:	no
Other arguments:	none

mmview

Invokes the multimedia editor to read the selected documents. If you select multiple documents, all the documents will be read into the editor. You may view each one by using the **Editor-Buffer-Switch To** command.

Requires window system:	yes
Other arguments:	none

6.2.3 Other Multimedia Utility Functions

This section gives a brief description of several utility programs that convert text files into multimedia documents, allow you to incorporate text-only electronic mail messages into your multimedia mailbox, and to be alerted when you have received new mail.

mmify

Converts a text file into a BBN/Slate multimedia document. BBN/Slate interprets the text file as verbatim text. It preserves line breaks and indentation. If the first line of the file contains a colon (:) before the first space, BBN/Slate assumes the file is a text mail message with an initial mail header. It assumes that the header conforms to the specification in DARPA Internet Request For Comments number 822 (RFC822). BBN/Slate extends this header and converts it into BBN/Slate document headers.

The `mmify` program can read from a file or from its standard input. It writes the generated BBN/Slate document on its standard output. Refer to the UNIX on-line manual page for more information.

mmdeliver

Part of the BBN/Slate electronic mail system. It is used to detect multimedia mail messages sent to you by other users, and to deliver them to your multimedia mailbox. It can also be used to automatically convert text messages into multimedia messages, automatically file messages in folders for you, and automatically discard messages before you see them.

The `mmdeliver` program is normally installed and maintained by a system administrator. For more information, see Chapter 3.

mmwatch

Shell script that controls the appearance of an icon on the screen. If you have new BBN/Slate mail, the icon shows a letter in an inbox, if not, the icon is an empty inbox. You can use `mmwatch` to watch for new mail if your workstation does not have enough memory to run a copy of the Document Manager all the time.

mmsnap

Captures a portion of the screen as a rasterfile. When you invoke `mmsnap`, it freezes the screen and you can specify the area you want to capture with the pointer. The screen image for the area you select is written to standard output using the Sun rasterfile format. You can redirect the output to a file that can be included in a multimedia document by using the Document Editor's **Add Rasterfile** command, or pass it through the `raster2bit` utility to create a BBN/Slate image file.

raster2bit

Converts Sun rasterfiles to RFC 797 bitmaps, the native bitmap format used by the Document Editor. The `raster2bit` utility reads a monochrome Sun rasterfile on standard input and outputs an RFC 797 format bitmap file on standard output. The output file has the extension `.bit` or `.bitmp`.

6.2.4 Support Commands

This section briefly describes five commands that perform utility functions for other parts of the BBN/Slate system. See the UNIX on-line manual pages for complete descriptions of these commands.

mmencode

Translates a binary BBN/Slate document into a compressed ASCII format so that it can be submitted to the local text mailer for delivery. BBN/Slate uses ASCII text mail transport facilities to send mail from one workstation to others. Because BBN/Slate documents contain some parts that are represented in binary form, messages must be encoded in an ASCII format before they can be mailed.

mmdecode	Reverts the encoding performed by the `mmencode` command, restoring an encoded document to its original multimedia form.
mmspell	Used by the **Editor-Document-Check Spelling** command in the Document Editor. When you check the spelling of a document, BBN/Slate extracts each word in turn and uses this program to determine if the word is spelled correctly.
mmpreview	Used by the **Editor-Print-Preview** commands in the Document Editor. It reads a file of PostScript code that represents a BBN/Slate document, and shows you what the document will look like when it is printed.
mmextract	Extracts the document headers from a message, and prints them on its standard output. This can be useful if you are writing your programs or shell scripts to manipulate BBN/Slate multimedia documents.

BBN Software Products
A Division of Bolt Beranek and Newman Inc.

Your opinions about our products and services are important to us. Please use this self-addressed, stamped form to help us evaluate this manual. Your response to the questions listed below will help us assess our current documentation and plan ways to improve it to meet your needs. Just fill in your responses, detach this sheet, fold and tape it, and drop it in the mail. We'll pay careful attention to what you say!

Manual Title: *BBN/Slate System Topics (Release 1.1)*

List your operating system: ______________________ Date: _____/_____/_____

General Comments:	**Yes**	**Somewhat**	**No**
Do you like this manual?	☐	☐	☐
Is it easy to read?	☐	☐	☐
Is the order of topics easy to follow?	☐	☐	☐
Is the information accurate?	☐	☐	☐
Can you easily find the information you need?	☐	☐	☐
Do the examples and illustrations help?	☐	☐	☐
Can you easily apply the described features to your own needs?	☐	☐	☐
Do you use on-line help?	☐	☐	☐

Comments and Suggestions (include chapter or page number where applicable):

How do you primarily use this manual?

☐ Introduction ☐ Tutorial ☐ Reference

☐ **Check here to receive information about our Education Services.**

☐ **Check here to receive the *BBN Software Quarterly* newsletter.**

Please fill in the following information:

Name	______________	Title	______________
Company	______________	Street	______________
City	______________	State	______________
Zip Code	______________	Phone	______________

Cut or fold along dotted line

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

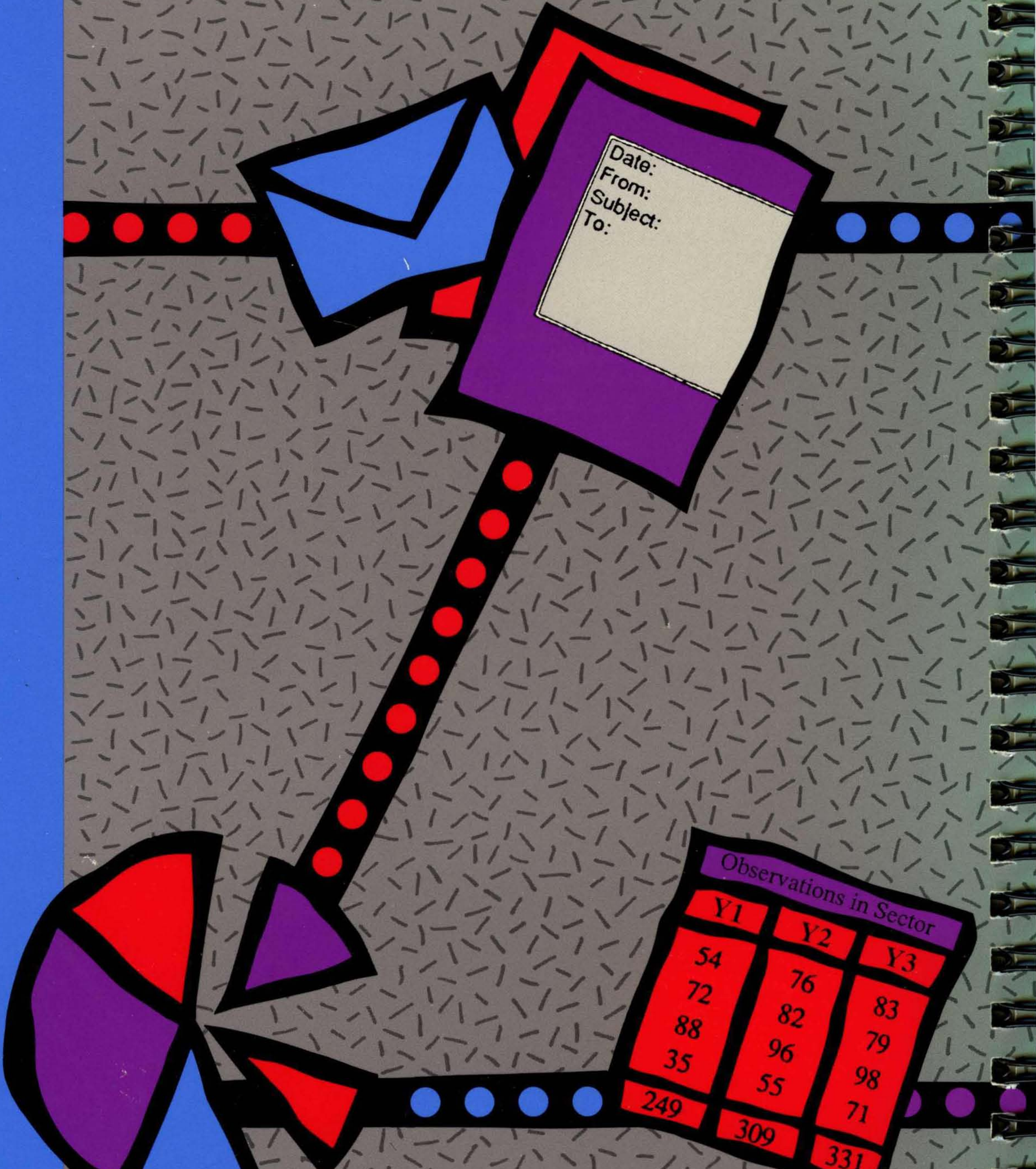
BUSINESS REPLY ENVELOPE

FIRST CLASS PERMIT NO. 36450 CAMBRIDGE, MA

POSTAGE WILL BE PAID BY ADDRESSEE

BBN Software Products
A Division of Bolt Beranek and Newman Inc.
10 Fawcett Street
Cambridge, MA 02138

Attn: Manager, Documentation



2523011

BBN Software Products
A Division of Bolt Beranek and Newman Inc.