

## Table of Contents

	Page
<b>I SYMBOLICS CONCORDIA 3.1 RELEASE NOTES</b>	<b>5</b>
1 Overview of Symbolics Concordia 3.1	7
2 Summary of New Commands for Symbolics Concordia 3.1	11
<b>II SYMBOLICS CONCORDIA INSTALLATION GUIDE</b>	<b>13</b>
3 Symbolics Concordia Installation Procedure	15
4 Documentation Database Compression and Decompression	17
4.1 What is Documentation Database Compression?	17
4.2 Why Decompress the Documentation Database?	17
4.3 Can Decompressing the Documentation Database Be Skipped?	18
4.4 Decompression Procedure	18
<b>III SYMBOLICS CONCORDIA WORKBOOK</b>	<b>19</b>
5 Symbolics Concordia Workbook: Creating a Record	23
5.1 Selecting the Symbolics Concordia Editor	23
5.2 Creating a File for Your New Record	25
5.3 Creating the Record	26
5.4 Looking at the Record Template	26
5.5 Putting Some Text in the Record	27
5.6 Filling in the Oneliner and Keywords Fields	29
5.6.1 Filling in the Oneliner Field	29
5.6.2 Filling in the Keywords Field	31
5.7 Printing the Record	32
6 Symbolics Concordia Workbook: Creating a Short Document	33
6.1 Planning Your Document	33
6.2 Creating Records for the Topics in Your Outline	33
6.3 Filling in the Contents of Your Records	35
6.4 Changing the Title of a Record	37
6.5 Adding Formatting Markup to Your Records	38
6.6 Collected Record Names Pane	39
6.7 Linking the Records	39
6.8 Looking At the Document	41
7 Symbolics Concordia Workbook: Creating Graphics	43

7.1	Creating a Diagram	43
7.2	Entering the Diagram Into a Record	46
7.3	Creating a Screen Snapshot	48
7.4	Entering the Screen Snapshot Into a Record	50
<b>8</b>	<b>Symbolics Concordia Workbook: Viewing Your Document in the Page Previewer</b>	<b>53</b>
<b>IV</b>	<b>SYMBOLICS CONCORDIA WRITER'S GUIDE</b>	<b>57</b>
<b>9</b>	<b>Introduction to the Symbolics Concordia Writer's Guide</b>	<b>59</b>
9.1	The Symbolics Concordia Window	59
9.1.1	Customizing the Symbolics Concordia Window	61
<b>10</b>	<b>Creating Documents with Symbolics Concordia</b>	<b>65</b>
10.1	Using Documentation Records	65
10.2	Getting Started in Symbolics Concordia	68
10.3	Getting Help in Symbolics Concordia	70
<b>11</b>	<b>Creating and Filling in a Documentation Record</b>	<b>71</b>
11.1	Symbolics Concordia Record Fields	73
<b>12</b>	<b>Managing Documentation Records</b>	<b>79</b>
12.1	Editing the Name of a Symbolics Concordia Record	79
12.1.1	Renaming a Record	79
12.1.2	Changing the Capitalization and Character Style of a Record Topic	80
12.1.3	Changing the Way a Record Topic Displays	80
12.2	Locating Records	81
12.3	Moving Records	81
12.4	Deleting Documentation Records	83
12.5	Viewing Records	83
<b>13</b>	<b>Building a Document: Linking Records</b>	<b>85</b>
13.1	Link Types	86
13.1.1	Using Links to Incorporate Text	86
13.1.2	Using Links to Refer to Other Records	89
13.1.3	Using Links to Create Tables	92
13.2	The Significance of Links	95
13.2.1	How the Formatter Processes a Record	95
13.2.2	Hierarchy of a Book	97
13.2.3	Online Reading Path	102
13.3	Using the Collected Record Names Pane	104
13.4	Link Commands	105
<b>14</b>	<b>Examining and Changing the Organization of a Document</b>	<b>109</b>

14.1	Examining the Organization of Your Document	109
14.2	Changing the Organization of Your Document	110
<b>15</b>	<b>Controlling Your Document's Appearance</b>	<b>113</b>
15.1	Introduction to Symbolics Concordia Markup Language	113
15.1.1	Environments and How to Use Them	113
15.1.2	Markup Commands and How to Use Them	118
15.1.3	Internal and External Markup	119
15.2	Concordia Markup Language	119
15.2.1	Some Commonly Used Environments	119
15.2.2	The Most Common Environment Attributes	137
15.2.3	The Most Common Markup Commands	138
15.2.4	Specifying Page Headings and Footings	139
15.2.5	Dictionary of Symbolics Concordia Markup Environments and Commands	141
15.2.6	List of Symbolics Concordia Attributes	170
15.3	Character Styles in Symbolics Concordia	174
15.4	Marking up Lisp Language Objects	175
<b>16</b>	<b>Moving Around Symbolics Concordia</b>	<b>177</b>
16.1	Moving Around the Symbolics Concordia Buffer	177
16.2	Moving Among the Symbolics Concordia Editor, Page Previewer, Graphic Editor, and Book Design Browser	178
<b>17</b>	<b>Inserting Illustrations in Your Symbolics Concordia Documents</b>	<b>179</b>
17.1	Putting Pictures in Text	179
17.1.1	Specifying the Size of Pictures	180
17.1.2	Tips and Techniques for Using Pictures in Documents	182
17.2	Creating Active Examples	183
17.3	Making Pictures Using Lisp	184
<b>18</b>	<b>Creating an Index in a Symbolics Concordia Document</b>	<b>187</b>
<b>19</b>	<b>Customizing Your Symbolics Concordia Documents</b>	<b>189</b>
19.1	Setting Document Formatting Options in Symbolics Concordia	190
19.1.1	Using Sage Static Variables to Set Document Formatting Options	192
19.1.2	Using Sage System Variables to Set Document Formatting Options	193
19.2	Concordia Example: A Book with Multiple Versions	193
<b>20</b>	<b>Displaying, Reviewing, and Publishing Documentation</b>	<b>197</b>
20.1	Displaying and Reviewing Documentation Online	197
20.1.1	Reviewing Work in the Symbolics Concordia Editor	197
20.1.2	Reviewing Work in Document Examiner	197
20.1.3	Proofreading Your Work	198

20.1.4	Displaying Documentation	198
20.1.5	Printing Draft Copies of a Document	199
20.2	Publishing Your Documentation	199
<b>21</b>	<b>Using the Page Previewer</b>	<b>201</b>
21.1	Overview of the Page Previewer	201
21.2	Producing a Document as a Book	201
21.3	Formatting Pages	202
21.4	Resolving Crossreferences	202
21.5	Moving to Another Page	204
21.6	Fixing Formatting Errors	205
21.7	Displaying Previously Formatted Pages	205
21.8	Printing Formatted Pages	205
21.9	Page Previewer Commands	206
<b>22</b>	<b>Workstyle Issues in Writing with Symbolics Concordia</b>	<b>211</b>
22.1	Introduction to Modular Writing	211
22.2	An Example of Writing for Modularity	211
22.3	Using Crossreferences in Modular Writing	212
22.4	Problems and Solutions for Modular Writing	213
<b>23</b>	<b>Symbolics Concordia Customizations for Your Init File</b>	<b>215</b>
23.1	Customizing the Editor Pane	215
23.2	Setting the Lookup Mode in the Document Examiner	215
23.3	Setting the Buffer Locking Mode in Symbolics Concordia	216
<b>24</b>	<b>Recovering From Errors in Symbolics Concordia</b>	<b>217</b>
24.1	When Symbolics Concordia Misplaces the Cursor	217
24.2	When Markup Damages the Undo History	218
<b>25</b>	<b>Dictionary of Symbolics Concordia Editor Commands</b>	<b>219</b>
<b>26</b>	<b>Symbolics Concordia Reference Card</b>	<b>231</b>
<b>V</b>	<b>USING THE GRAPHIC EDITOR</b>	<b>233</b>
<b>27</b>	<b>Overview</b>	<b>235</b>
<b>28</b>	<b>Getting Started with the Graphic Editor</b>	<b>237</b>
28.1	The Graphic Editor Screen Layout	237
28.2	Using the Mouse in the Graphic Editor	238
28.3	Changing Graphic Editor Defaults	240
28.4	Controlling Mouse-Sensitivity in Graphic Editor Drawings	245
<b>29</b>	<b>Creating Shapes with the Graphic Editor</b>	<b>249</b>
<b>30</b>	<b>Creating Drawings with the Graphic Editor</b>	<b>259</b>

<b>31</b>	<b>Managing Drawings</b>	<b>265</b>
31.1	Changing the View of Drawings	265
31.2	Copying Drawings	266
31.3	Storing Drawings in Files	267
<b>32</b>	<b>Using Drawings in Symbolics Concordia Documents</b>	<b>269</b>
32.1	Inserting a Bitmap Image Into a Graphic Editor Drawing	269
32.2	Including a Graphic Editor Drawing in a Document	270
<b>33</b>	<b>Strategies and Hints for Using the Graphic Editor</b>	<b>273</b>
33.1	Creating Non-Standard Shapes	273
33.1.1	Drawing Irregular Shapes	273
33.1.2	Combining Entities to Produce Complex Shapes	275
33.1.3	Using Filling to Produce Shapes	277
33.2	Grouping Entities	279
33.3	Selecting Entities	280
33.4	Transformations on Entities	281
33.5	Controlling the Size and Alignment of Entities	283
33.6	Filling a Shape Entity with a Pattern	285
33.7	Example of Using the Graphic Editor	286
<b>34</b>	<b>Dictionary of Graphic Editor Commands</b>	<b>293</b>
<b>35</b>	<b>Using the Bitmap Editor</b>	<b>317</b>
35.1	Using the Bitmap Editor Window	317
35.2	Entering Bitmap Editor Commands	318
35.3	Bitmap Editor Basic Concepts	319
35.3.1	Drawing	319
35.3.2	Using the Gray and Black Planes to Edit Images	320
35.3.3	Using Registers to Copy Images	321
35.3.4	Viewing Bitmap Images	321
35.4	Using Screen Images as Illustrations	322
35.4.1	Capturing Screen Images	322
35.4.2	Inserting a Bitmap Image Into a Graphic Editor Drawing	325
35.4.3	Editing Bitmap Images	326
35.5	Dictionary of Bitmap Editor Commands	327
<b>36</b>	<b>Using the Stipple Editor</b>	<b>335</b>
36.1	Creating a New Stipple Pattern	335
36.2	Techniques for Using the Stipple Editor	336
36.3	Dictionary of Stipple Editor Commands	336
<b>VI</b>	<b>CONVERTING EXISTING DOCUMENTATION TO CONCORDIA</b>	<b>341</b>
<b>37</b>	<b>Introduction</b>	<b>343</b>

<b>38</b>	<b>Converting Scribe Documents to Symbolics Concordia</b>	<b>345</b>
<b>39</b>	<b>Converting Text Files to Symbolics Concordia</b>	<b>347</b>
<b>40</b>	<b>Converting Graphics to Symbolics Concordia</b>	<b>349</b>
<b>41</b>	<b>Quick Conversion for Producing Printed Books</b>	<b>351</b>
<b>42</b>	<b>Converting Documents From Concordia to Scribe</b>	<b>353</b>
<b>VII</b>	<b>SYMBOLICS CONCORDIA ADMINISTRATOR'S GUIDE</b>	<b>355</b>
<b>43</b>	<b>Guide to Symbolics Concordia Book Design</b>	<b>357</b>
43.1	Introduction to Symbolics Concordia Book Design	357
43.1.1	What is Book Design?	357
43.1.2	Book Design Vocabulary	357
43.1.3	Inheritance Dependency	359
43.1.4	Modifying the Appearance of Your Documentation	361
43.2	How the Book Design Sources Are Organized	362
43.2.1	Book Design Source Files	362
43.2.2	Document Device Types	363
43.2.3	Book Design Elements	366
43.3	Book Design Browser	369
43.3.1	Introduction to the Book Design Browser	369
43.3.2	Layout of the Book Design Browser Window	370
43.3.3	Getting Help in the Book Design Browser	371
43.3.4	Issuing Commands in the Book Design Browser	372
43.3.5	Mouse Sensitivity in the Book Design Browser	373
43.4	Modifying a Document Device Type	373
43.4.1	Introduction	373
43.4.2	Examples of Document Device Type Modifications	374
43.5	Modifying a Single Book	384
43.5.1	Specifying Page Headings and Footings	385
43.5.2	Handling Tables of Contents and Frontmatter	387
43.5.3	Creating Title Pages	388
43.6	Modifying a Single Instance of an Environment	389
43.7	Creating a New Document Type	390
43.7.1	Creating a Letter Document Type	393
43.7.2	Creating Your Own Crossreference Appearance	399
43.7.3	Formatting Tables of Contents and Lists of Figures	401
43.7.4	Formatting Page Headings	403
43.8	Creating Your Own 3Symanual Document Type	404
43.9	Creating an Article Document Type	405
43.9.1	Defining Counters for an Article	405
43.9.2	Defining Headings for an Article	408
43.10	Defining a Book Design System for a Site	410
43.11	Dictionary of Book Design Browser Commands	412

43.11.1	Clear Display	412
43.11.2	Describe Book Design Element	412
43.11.3	Describe Document Device Type	413
43.11.4	Describe Environment	414
43.11.5	Edit Book Design Element	416
43.11.6	List Book Design Elements Using Environment	416
43.11.7	Show Dependencies on Book Design Element	416
43.11.8	Show Dependencies on Environment	417
43.12	Book Design Functions Dictionary	418
43.12.1	Internally Used Formatting Styles	418
43.12.2	Internally Used Heading Styles	420
43.12.3	Internally Used Table of Contents Entry Styles	421
43.12.4	Book Design Functions and Variables	421
43.12.5	Sectioning Commands for Conversion Compatibility	429
<b>44</b>	<b>Guide to Documentation System Maintenance</b>	<b>431</b>
44.1	Setting up Your Documentation System	431
44.1.1	Compiling	431
44.1.2	The System Declaration for a Documentation System	431
44.1.3	Registering Your Documentation System	435
44.1.4	Updating the System Declaration	435
44.1.5	Packages for Documentation Systems	435
44.1.6	A System Declaration for a Large Document Set	436
44.2	Compiling a Documentation System	438
44.2.1	Making Sure the System Declaration is Correct	439
44.2.2	The Actual Compilation	439
44.2.3	Reap Protecting the New Version	440
44.2.4	Compiling a Documentation System for Multiple Machine Types	440
44.2.5	Summary of Compiling the Documentation Database	442
44.2.6	Patching a Documentation System	442
44.3	Setting a System's Status	445
44.3.1	System Status in the Herald	445
44.3.2	System Status for Loading and Other Operations	446
44.4	Maintaining Your Documentation Sources	446
44.4.1	Cleaning up	446
44.4.2	Checking Your Documentation System	448
44.5	The Journal Directories	450
44.6	Distributing Documentation Systems	451
44.6.1	Distributing NSage Patches	451
44.6.2	Distributing Book Designs	452
44.6.3	Quality Assurance for Documentation Systems	452
44.6.4	Replacing Symbolics Documentation with Your Own Documentation System	452
44.6.5	Distributing Documentation for Macintosh Document Examiner	453

<b>VIII APPLICATION PROGRAMMERS' INTERFACE GUIDE TO SYMBOLICS CONCORDIA</b>	<b>457</b>
<b>45 Displaying Documentation Under Application Program Control</b>	<b>459</b>
<b>46 Finding Topics Under Application Program Control</b>	<b>461</b>
<b>47 Displaying Overviews Under Application Program Control</b>	<b>463</b>
<b>48 Creating Records Under Application Program Control</b>	<b>465</b>
<b>49 Displaying Graphics Under Application Program Control</b>	<b>467</b>
<b>50 Creating Mouse Sensitivity in Your Output</b>	<b>469</b>
<b>51 Displaying Formatted Text Under Application Program Control</b>	<b>471</b>
51.1 Formatting From a Source with Embedded Formatting Commands	471
<b>52 Application Programmers' Dictionary</b>	<b>473</b>
<b>53 Replacing Symbolics Documentation with Your Own Documentation System</b>	<b>479</b>



## List of Figures

	Page
1 Symbolics Concordia Editor	24
2 New .sab File with an Attribute List	25
3 Record Fields	27
4 Record with Text	28
5 Record with the Oneliner Field Filled In	30
6 Oneliner Field Displayed in Document Examiner	30
7 A Record with Its Keywords Field Filled In	32
8 The Keywords Field Displayed in Document Examiner	32
9 Record with Record-Name Field Added	38
10 Displayed Record with Markup	39
11 Collected Record Names Pane	39
12 Record with Links	41
13 The Graphic Editor	44
14 Example Diagrams	46
15 The FUNCTION Q menu	48
16 Page Previewer	54
17 A Formatted Document in the Page Previewer	55
18 The authoring, printing, and viewing components all rely on the centralized database.	65
19 The Concordia window	69
20 Concordia inserts a template for a new record.	73
21 Oneliner and Keywords fields are used in the Overview in Document Examiner	75
22 A Filled-in Record	77
23 Links establish relationships among records.	85
24 Include links inserted in a record.	86
25 View produced by displaying include links.	87
26 A contents link inserted in a record.	88
27 View produced by displaying a contents link.	89
28 Crossreference links inserted in a record.	90
29 View produced by displaying crossreference links.	90
30 Precis links inserted in a record.	93
31 View produced by displaying precis links.	93
32 Precis links embedded in a Description environment.	94
33 A Description environment changes the appearance of the precis link view.	94
34 Precis links display the argument list of Lisp functions.	94
35 Precis links create a mouse-sensitive table of contents.	95
36 How the formatter processes links.	96
37 The table of contents shows the structure of a book.	97

38	The script record sets up the structure of your book.	100
39	A sample chapter-level record.	100
40	Diagram of book structure.	101
41	A menu appears when you edit a link type or target.	111
42	Mouse	180
43	Click on the page holders to move from page to page.	204
44	Selected Entities	235
45	Moving the Selected Entities	235
46	The Graphic Editor Screen	238
47	Text Transformation Options	256
48	Tree	276
49	The Bitmap Editor Screen	318
50	FUNCTION Ø □	323
51	Book Design Browser Icon	370
52	Initial Book Design Browser	370
53	Book Design Browser Icon	374
54	Describe Environment Itemize GENERIC GENERIC	375
55	Itemize for Generic Generic Document Device Type with the Spread Attribute Modified.	377
56	Describe Environment Enumerate GENERIC GENERIC	378
57	Enumerate for Generic Generic Document Device Type with the Spread Attribute Modified.	380
58	Describe Document Device Type 3symanual LGP2	381
59	Describe Book Design Element 3symanual-lgp2	382
60	Modify Markup modifying Description environment.	385
61	The menu shows the current attributes of an Itemize environment.	390
62	Modified Environment Markup showing telltale ellipsis.	390
63	Describe Book Design Element user-visible-environments	412
64	Describe Book Design Element 3symanual-lgp2	413
65	Describe Document Device Type 3symanual LGP2	414
66	Describe Environment Example 3symanual LGP2	415
67	List Book Design Elements Using Environment Description	416
68	Show Dependencies On Book Design Element user-visible-environments	417
69	Show Dependencies On Environment Itemize	417
70	<b>sage:graph-book-design-users</b>	426
71	<b>sage:graph-book-design-uses</b>	427
72	Document Sets.	454
73	The Macintosh desktop.	456
74	Click on me to select <i>Symbolics Concordia</i> .	469

## Preface to Symbolics Concordia

### Overview of Symbolics Concordia Documentation

The document is divided into these sections:

- The Symbolics Concordia 3.1 Release Notes describes the features for this release.
- The Symbolics Concordia Installation Guide describes procedures for installing Symbolics Concordia at your site.
- The Symbolics Concordia Workbook provides a tutorial introduction to document creation and production. It includes exercises in using the Symbolics Concordia text editor, the Page Previewer, and the Graphic Editor.
- The Symbolics Concordia Writer's Guide fully describes the Symbolics Concordia text editor, the Symbolics Concordia Page Previewer, and workstyle issues related to writing and producing modular documentation.
- Using the the Graphic Editor describes how to use the Graphic editor, an object-based drawing program that lets you create illustrations, diagrams, and other graphics. This includes a description of how to use the Bitmap and Stipple Editors for editing screen images and for creating stipple patterns, respectively.
- Converting Existing Documentation to Concordia describes tools for importing and converting documents into Symbolics Concordia. This includes procedures for converting documents done in Scribe (or in old Symbolics writer tools).
- Symbolics Concordia Administrator's Guide includes procedures for creating your own book designs, and procedures for documentation system maintenance.
- The Application Programmer's Interface Guide describes how to use features of the application development substrate with Symbolics Concordia. These features include Dynamic Windows, the presentation substrate, **dw:define-program-framework**, and the generic graphics substrate.

### Documentation Conventions

This documentation uses the following notation conventions:

<b>cond</b> , <b>zl:hostat</b>	Printed representation of Lisp objects in running text.
RETURN, ABORT, c-F	keys on the Symbolics Keyboard.
SPACE	Space bar.
login	Literal typein.
(make-symbol "foo")	Lisp code examples.

<b>(function-name</b> <i>arg1</i> &optional <i>arg2</i> )	Syntax description of the invocation of <b>function-name</b> .
<i>arg1</i>	Argument to the function <b>function-name</b> , usually expressed as a word that reflects the type of argument (for example, <i>string</i> ).
&optional	Introduces optional argument(s).
Show File, Start	Command Processor command names and Front-end Processor (FEP) command names appear with the initial letter of each word capitalized.
<i>m-x</i> Insert File, Insert File ( <i>m-x</i> )	Extended command names in Zmacs, Zmail, and Symbolics Concordia appear with the <i>m-x</i> notation either preceding the command name, or following it in parentheses. Both versions mean press <i>m-x</i> and then type the command name.
[Map Over]	Menu items. Click Left to select a menu item, unless other operations are indicated. (See the section "Mouse Command Conventions" in <i>Genera 8.0 Reference Cards</i> .)
Left, Middle, Right	Mouse clicks.
<i>sh-Right</i> , <i>c-m-Middle</i>	Modified mouse clicks. For example, <i>sh-Right</i> means hold down the SHIFT key while clicking Right on the mouse, and <i>c-m-Middle</i> means hold down CONTROL and META while clicking Middle.

## Modifier Key Conventions

Modifier keys are designed to be held down while pressing other keys. They do not themselves transmit characters. A combined keystroke like META-*x* is pronounced "meta x" and written as *m-x*. This notation means that you press the META key and, while holding it down, press the *x* key.

Modifier keys are abbreviated as follows:

CONTROL	<i>c-</i>
META	<i>m-</i>
SUPER	<i>s-</i>
HYPER	<i>h-</i>
SHIFT	<i>sh-</i>
SYMBOL	<i>sy-</i>

Modifier keys can be used in combination, as well as singly. For example, the notation *c-m-y* indicates that you should hold down both the CONTROL and the META keys while pressing *y*.

Modifier keys can also be used, both singly and in combination, to modify mouse commands. For example, the notation *sh-Left* means hold down the SHIFT key

while clicking Left on the mouse and `c-m-Middle` means hold down CONTROL and META while clicking Middle.

The keys with white lettering (like `X` or SELECT) all transmit characters. Combinations of these keys should be pressed in sequence, one after the other (for example, `SELECT L`). This notation means that you press the SELECT key, release it, and then press the L key.

LOCAL is an exception to this rule. Despite its white lettering, you must hold it down while pressing another key, or it has no effect. For example, to brighten the image on your monitor, you would hold down LOCAL while pressing B.



## **PART I.**

# **SYMBOLICS CONCORDIA 3.1 RELEASE NOTES**





## 1. Overview of Symbolics Concordia 3.1

Symbolics Concordia 3.1 provides the following enhancements:

- **Mouse-Sensitive Text and Graphic Entities**

You can now create mouse-sensitive text and graphic entities by associating them with a presentation type and a presentation object.

- The presentation type (a documentation topic, a Lisp form, a path-name, a CP command, or none) determines where and when a text or graphic entity is mouse-sensitive, and defines what kinds of presentation objects are valid.
- The presentation object (the thing) controls what happens when you click on the entity.

Use the "Make Active Text Command" to associate text in your Symbolics Concordia buffer with a presentation type and a presentation object. When you click on the text in a document, the associated presentation object is displayed (according to rules determined by its presentation type).

Use the "Change Entity Graphic Editor Command" or the "Edit Defaults Graphic Editor Command" to associate presentation types and presentation objects with Graphic Editor drawing entities. For more information, see the section "Controlling Mouse-Sensitivity in Graphic Editor Drawings", page 245.

- **Dynamic Text**

A new Concordia command (`DynamicText`) provides a way to use the programmatic features of Concordia in your document. The `DynamicText` command calls a function to produce text and formatter directives that are inserted into the document when it is formatted. `DynamicText` accepts as its argument the name of a Lisp function to call.

This means that a Symbolics Concordia record that includes `DynamicText` commands can format differently, depending upon variables or other state in the machine at the time it is formatted.

- **Conditional Document Formatting**


Writers can now define document formatting options that can be set in the document itself, or can be set later by the reader when the document is viewed online or formatted. Using these tools, readers of the document can choose formatting options that meet their immediate information needs.

- A new Symbolics Concordia Case environment allows you to define document formatting options.
- You can set document formatting options in the the Concordia Editor, the Page Previewer, the Document Examiner, or in a Lisp listener.

See the section "Customizing Your Symbolics Concordia Documents", page 189.

- **Book Design Browser**

The Book Design Browser is a set of tools that you can use to do book design more easily.

You can select the Book Design Browser by clicking on  **B** at the top of the Symbolics Concordia screen. See the section "Guide to Symbolics Concordia Book Design", page 357.

- **New Tools for Converting Existing Documents to Concordia**

Convert Flat Text to Record Command

Creates a new record from the marked region.

Clear Record Name Prefix Command

Clears the record name prefix prepended to newly created records.

Clear Record Name Suffix Command

Clears the record name suffix appended to newly created records.

Set Record Name Prefix Command

Sets the record name prefix prepended to newly created records.

Set Record Name Suffix Command

Sets the record name suffix appended to newly created records.

See the section "Converting Existing Documentation to Concordia", page 342.

- **New Editor Command Menu Display Options**

You can now click on topics in the Symbolics Concordia Editor command menu to expand them into a mouse-sensitive list of the related commands. (Click again on a topic to contract it.)

- **Enhanced Graphic Editor Documentation**

Graphic, Bitmap, and Stipple Editor documentation now includes more extensive user information, including more examples and complete command dictionaries. See the section "Using the Graphic Editor", page 234.

- **New Dictionary of Symbolics Concordia Markup Commands and Environments**

This dictionary documents all Symbolics Concordia environments and commands. Most command and environment command descriptions now include examples. See the section "Dictionary of Symbolics Concordia Markup Environments and Commands", page 141.

- **Miscellaneous**

Flood Region Bitmap Editor command now works properly.

Concordia Modify environment now works.

Read and Write Image File Commands are now imported into the Graphic, and Bitmap Editors.



## 2. Summary of New Commands for Symbolics Concordia 3.1

These commands are new or have been significantly improved for this release:

- Case Command      Defines Symbolics Concordia document formatting options.
- Change Entity      Changes the display characteristics of one or more entities.
- Clear Record Name Prefix Command  
Clears the record name prefix prepended to newly created records.
- Clear Record Name Suffix Command  
Clears the record name suffix appended to newly created records.
- Clear Sage Variable Command  
Clears the named sage variable.
- Convert Flat Text to Record Command  
Creates a new record from the marked region.
- Copy Drawing to Image Graphic Editor Command  
Makes a bitmap image of the current Graphic Editor drawing. The first argument is the name of the new image. The second argument is a scale factor (the default of 1 indicates full size).
- Count Records in Buffer Command  
Displays the number of records in the current Symbolics Concordia buffer.
- DynamicText Command  
Calls a function to produce text and formatter directives that are inserted into the document when it is formatted.
- Edit Defaults      Changes the default drawing parameters to be used for subsequently created entities.
- Format Pages Page Previewer Command  
Formats and displays the specified record.
- List Sage Variables Command  
Displays a list of the sage variables that are currently set.
- Make Active Text Command  
Creates mouse sensitive text regions in Concordia documents.
- PermanentString Command  
PermanentString value assignments set document formatting options that persist across formatting runs. (String assignments go away when the formatter finishes).

**sage::register-book** *topic-string &key (:mnemonic "") :document-type :highest-structural-level :cover :remarks :symcopy :doc# :effectivedate :releaseversion :marketing :mitcopy :authorgroup :design :cover-printer :text-printer :printer :confidential :doctrademarks :deferred-home :sage-variables*  
 Registers a Symbolics Concordia topic as a book and specifies its design.

**Set Record Name Prefix Command**

Sets the record name prefix prepended to newly created records.

**Set Record Name Suffix Command**

Sets the record name suffix appended to newly created records.

**Set Sage Variable Command**

Sets the named sage variable to the specified string value. Use sage variables as Case selectors to control document formatting options.

**Value Command**

Retrieves and prints the value of its argument, which should be a counter or a string defined by the String command.

## **PART II.**

# **SYMBOLICS CONCORDIA INSTALLATION GUIDE**





### 3. Symbolics Concordia Installation Procedure

1. Boot a site-configured Genera 8.1 world.
2. Load the IP-TCP software:
 

```
Load System IP-TCP
```
3. Restore the contents of the Symbolics Concordia tape, as follows:
  - a. Place the Symbolics Concordia distribution tape in the tape drive.
  - b. Issue this command to the Command Processor:
 

```
Restore Distribution
```
  - c. Enter a tape spec at the prompt:
 

```
Enter a tape spec [default Local: Cart]:
```

If you are using the cartridge tape drive on the local host, just press RETURN. Otherwise, enter the host name and the drive type.

Load Symbolics Concordia, as follows (it takes approximately 20 minutes).

```
Load System Concordia :Query No
```

If you are loading Concordia on a 3600-family machine, you encounter an error that puts you in the debugger. You should simply resume. For more specific details, see the section "Loading Concordia on 3600-Family Machines", page 16.
4. Decompress the Symbolics document database. The purpose of database compression and decompression is explained in "Documentation Database Compression and Decompression". You should refer to this section to decide if you should decompress the documentation database. This process takes up to 40 minutes. Use the function **sage:load-doc-database-for-writer**.
 

```
(sage:load-doc-database-for-writer)
```
5. Reset the threshold at which the user receives warnings about address space being low. This is suggested because saving modified Symbolics Concordia buffers uses a lot of address space. The normal threshold for warnings is not adequate for protecting against running out of address space.
 

```
(setq si:gc-warning-threshold 4000000)
```
6. Save the results of your installation. See the section "Customizing and Saving Worlds" in *Site Operations*.
7. Load the Lock Simple system to establish concurrency control for writing out Symbolics Concordia files. All hosts that are to store Symbolics Concordia files must agree on the server host to which they send their Lock-Simple re-

quests. If the files to be locked are themselves resident on a particular host, it is a good idea to make that host a Lock-Simple server. If you use a server on another host, you have twice the risk that an unavailable host will make your files ununlockable, preventing you from saving them. If you are storing Symbolics Concordia files on a MacIvory, the Concordia files must be stored in a LMFS, they cannot be stored in the MacFS. To install Lock Simple:

- a. Load the system called Lock Simple on each file server that is going to store Symbolics Concordia files.

Load System Lock Simple

- b. Use the command Add Service to Hosts for each of these file servers to indicate that it is now a Lock Server:

Add Service to Hosts LOCK-SIMPLE CHAOS-TOKEN-LIST LOCK-SIMPLE *host-names*

See the section "Add Services to Hosts Command" in *Genera Handbook*.

- c. Do Enable Services LOCK-SIMPLE on each of the file servers that is now a Lock Server.

One host at your site should be designated as a surrogate Lock Server to process Lock-Simple requests for files on hosts that are not set up as Lock-Simple servers. To set up a surrogate Lock Server, use this procedure:

- a. Edit the namespace object for your site:

Edit Namespace Object Site *your-site*

- b. Add the user property pair

Lock-Simple-Server *host*

where *host* is the server which will be the surrogate Lock server.

- c. Do Enable Services LOCK-SIMPLE on the surrogate Lock server.

### **Loading Concordia on 3600-Family Machines**

When you load Concordia on a 3600-family machine, you encounter an error that puts you in the debugger with the following message:

Error: Attempt to define method

(:INTERNAL (FLAVOR:METHOD MTB:GRED-CAPTURE-PIXMAP...))

The file in question (SYS:GRAPHIC-EDITOR:PICT-FILES.BIN.1) shows up in your status line. You should skip loading this file if you are running Concordia on a 3600-family machine. You do not lose any functionality by not loading this file.

## 4. Documentation Database Compression and Decompression

### 4.1. What is Documentation Database Compression?

Online documentation resides on the disk in binary files. The indexing information for searching and lookup is always loaded into the world. In a world including Symbolics Concordia, the index information for the documentation records (unique ids and so on) is large, because each record may have several versions and other information associated with it. This information is organized by a structure called a *record group*. Each record can have a version of itself that is a *published record* (patched or compiled into the database) and an *edited record* (one that has been changed but not installed or patched yet). Symbolics Concordia keeps careful track of each version of the record in the record group. This adds quite a number of blocks to the world size. The majority of the information cached in a record group is of interest only when editing documentation. For users who aren't editing documentation, it is definitely advantageous to have all vestiges of anything other than the published record removed.

During the compilation of the Symbolics Documentation Database (see the section "Compiling", page 431), the index information loaded into the world is *compressed*. This means that all the .sab files included in the new version are loaded and the compiler removes the "group" from each record in the record registry. When this compression is finished, the information that was removed is stored on disk in two index files, SYS:DOC:INSTALLED-N;COMPRESSED-INDEX-A.BIN, and SYS:DOC:INSTALLED-N;COMPRESSED-INDEX-AW.BIN. where *n* is the version number of the new documentation database. Finally, a copy of each .sab file in the documentation system is written out to this installed-*n* directory. This provides a small directory of only those versions of the documentation files included in the current documentation database for lookup in Document Examiner.

**CAVEAT: At this time there can be only one compressed documentation database in a world. The Symbolics documentation is it.**

### 4.2. Why Decompress the Documentation Database?

You decompress the documentation database to be able to make crossreferences to Symbolics documentation and so that any documentation systems you create can be patched (incrementally updated; see the section "Patching a Documentation System", page 442).

When someone makes crossreferences to Symbolics documentation records, the complete record group information for those records must be made available. Similarly, when a documentation system is patched its complete record group information must be added to the index information in the world. This means that the structure to hold record group information must be in place in the world. *Decom-*

*pression* is the process that restores the structure, which was removed when the documentation database was compressed.

### 4.3. Can Decompressing the Documentation Database Be Skipped?

Most users of Symbolics Concordia should decompress the documentation database. However, if you are never going to make reference to Symbolics documentation in your own documentation, or to create a documentation system of your own to which you add patches (incrementally updates, see the section "Patching a Documentation System", page 442), you can skip the decompression step. This has the advantage of making your incremental world a little smaller. If you want the option to have your documentation refer to Symbolics documentation, or are planning to create your own patchable documentation system, you should decompress the database.

### 4.4. Decompression Procedure

After Symbolics Concordia is loaded (see the section "Symbolics Concordia Installation Procedure", page 15), run the function **sage:load-doc-database-for-writer**. This process takes up to 40 minutes. It uses the information stored on disk after the compression to construct a structure to hold the record groups. The record groups themselves are loaded in as a record is edited or reinstalled (temporarily or permanently by patching). The addition of this record group structure makes the world larger by about 10,000 blocks.

**sage:load-doc-database-for-writer** &key (:system-name "DOC") *Function*  
(:major-version **sage:\*doc-system-version\***)

Reverses the compression process done to the documentation database during recompilation. To do this, it loads the files `compressed-index-a.bin` and `compressed-index-aw.bin` from the installed documentation database directory and reconstructs the record group for each record in the documentation system. This allows tracking the edited and installed versions of each record for writers.

## **PART III.**

# **SYMBOLICS CONCORDIA WORKBOOK**

## **Preface to the Symbolics Concordia Workbook**

### **What is Symbolics Concordia?**

Symbolics Concordia is an integrated environment for creating, revising, publishing, and distributing documentation in both online and hardcopy form.

Symbolics Concordia is composed of the following:

- Symbolics Concordia text editor, a place to write your documentation.
- Page Previewer, a place to look at the format of your documentation.
- Graphic Editor, a place to create and manipulate illustrations, which can be inserted in text.
- Document Examiner, a place to view documentation online.
- A substrate for application programs to build upon.

Some general user-visible features of Symbolics Concordia are:

- You write text in *records*, rather than ordinary text files.
- Records are kept in files with a .sab file type, used only by Symbolics Concordia.
- Documentation files are not private; all writers at one work site can share a documentation database, which is composed of .sab files.
- Any documentation created with Symbolics Concordia, from a single record to a large document, can be viewed online in Document Examiner.

### **Purpose of this Workbook**

This workbook is intended to teach a writer how to use Symbolics Concordia to produce documents.

### **Who Should Use This Workbook?**

If you are a writer who is familiar with the Zmacs or EMACS text editors, this workbook is for you.

### **Who Should Not Use This Workbook?**

If you are not familiar with the Zmacs editor's commands, you will not be able to use this workbook. Suggestions for background reading are in the section "Prerequisites for the Symbolics Concordia Workbook".

## Prerequisites for the Symbolics Concordia Workbook

This workbook assumes that you are familiar with the information in the *Genera Workbook*. For example, you should be familiar with the Zmacs editor's keystrokes (for example, c-N), extended commands (commands that use m-X), mouse clicks (for example, c-Left) and basic file operations (saving and deleting files). If you do not know how to use these, we suggest you read the following sections in the *Genera Workbook*:

- "Workbook: Getting Around the System"
- "Workbook: Files and Buffers "
- "Workbook: Walk-through for Finding a File"
- "Workbook: Getting Familiar with Files"
- "Workbook: Document Examiner"
- "Workbook: More Zmacs"
- "Workbook: More Document Examiner"

Since the Symbolics Concordia text editor and the Zmacs text editor use many of the same commands, we suggest you read the following sections in *Editing and Mail* for a review of Zmacs commands. We list specific sections, since not all of the information in *Text Editing and Processing* is useful to Symbolics Concordia users.

In *Getting Started in Zmacs*, the following sections are useful:

- "Inserting Text in Zmacs"
- "Introduction to Moving the Cursor"
- "Killing and Yanking Text in Zmacs"
- "Creating and Saving Buffers and Files"
- "Introduction to Zmacs Commands"
- "Creating and Saving Buffers and Files"

In *Moving the Cursor in Zmacs*, the following sections are useful:

- "Motion Commands"
- "Motion by Paragraph"

In *Deleting and Transposing Text in Zmacs* the following sections are useful:

- "Deleting and Transposing Characters"
- "Deleting and Transposing Words"
- "Deleting and Transposing Lines"
- "Deleting and Transposing Lines"

In *Searching, Replacing, and Sorting in Zmacs*, all the sections are useful to users of Symbolics Concordia.

### **About the Symbolics Concordia Workbook**

The Symbolics Concordia Workbook is divided into four sessions. After completing all four sessions, you will know how to do the following:

- Create a record.
- Create a short document.
- Create graphics.
- Examine the format of a document in the Page Previewer.

In each session we discuss the concepts underlying the task presented in that session, but not in great detail. For a more in-depth discussion of the issues, use the reference documentation.

### **Workbook Organization**

The workbook sessions in this document contain these types of information:

*Concept sections* explain the ideas behind what you're trying to do.

*Instruction sections* show you the steps in a procedure.

*Workstyle suggestions* recommend ways to perform certain tasks to help you establish your own work habits.



## 5. Symbolics Concordia Workbook: Creating a Record

This first workbook session guides you through the steps in creating a single record.

### Concepts:

One of the major differences between Symbolics Concordia and documentation systems you might have used in the past is that in Symbolics Concordia you type text into *records*, rather than directly into files. Records are the basic building blocks of a document.

Think of each *record* as a unit that stores information about a single topic.

One of the questions you will ask yourself each time you write with Symbolics Concordia is: *How much and what type of information should I put in the record?*

The answer to this question is a judgment call, but it's easier if you think about how your record is going to be used. With Symbolics Concordia, you can produce two versions of a record: an *online* version and a *printed* version:

- The **online** version of your record will be read in Document Examiner. Document Examiner is a Genera facility where records can be read by anyone. The important point about reading records online is that the reader has the choice of reading a whole document, part of a document, or *one record at a time*. This means that your records will often be read independent of their context. You cannot count on the framework of your document to support the meaning of each record.
- The **printed** version of your record will be read as part of a document, and will be on paper. The reader of the printed version of a document is not aware that the document is a collection of individual records.

After you create a few records, you'll have a better idea about how much information should go into one.

### 5.1. Selecting the Symbolics Concordia Editor

The Symbolics Concordia Editor is where you create and revise documents.

#### Instructions:

1. Press `SELECT W` to select the Symbolics Concordia editor.

A window appears; it consists of several panes. For this session you'll need to know about two of these panes:

- The Symbolics Concordia Editor pane; this is the editor, where you create and fill in records.

- The Editor Commands Menu pane; where commands you use to create and manipulate documentation records are listed. You can click on topics in this menu to expand them into a list of their related mouse-sensitive commands. (Click again on the topic to contract it.)

Expand the topics containing the commands you use most frequently so that you can select them using the mouse. For more information, see the section "Symbolics Concordia Customizations for Your Init File", page 215.

Figure 1 shows the Symbolics Concordia editor window.

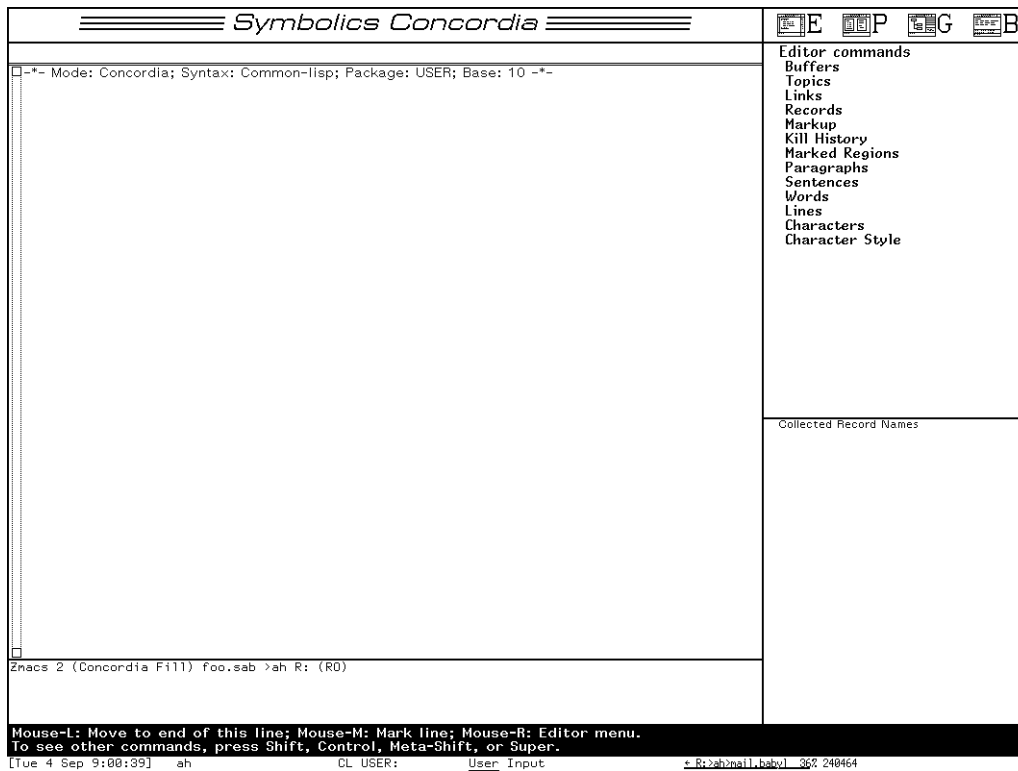


Figure 1. Symbolics Concordia Editor

## 5.2. Creating a File for Your New Record

Symbolics Concordia records are kept in files that have the .sab file type. (Files in turn are edited in buffers.) Before you can create a record, you must create a file to put it in. The file name must include the .sab extension.

**Note:** The initial buffer \*Concordia-1\* cannot be used for purposes of this tutorial. You must create a file of the type .sab before you can use Symbolics Concordia editor commands.

### Instructions:

1. Use `c-X c-F` to create a new file called `session-1.sab` in your default directory. The file will automatically be read into a buffer in your Symbolics Concordia pane. `c-X c-F` prompts you for a pathname. Here is an example:

```
Y:>rsw>lmdoc>session-1.sab
```

The new file has an attribute list at the top:

```
--Mode: Concordia; Syntax: Common-lisp; Package: USER; Base: 10--
```

Figure 2 shows a new .sab file with an attribute list.

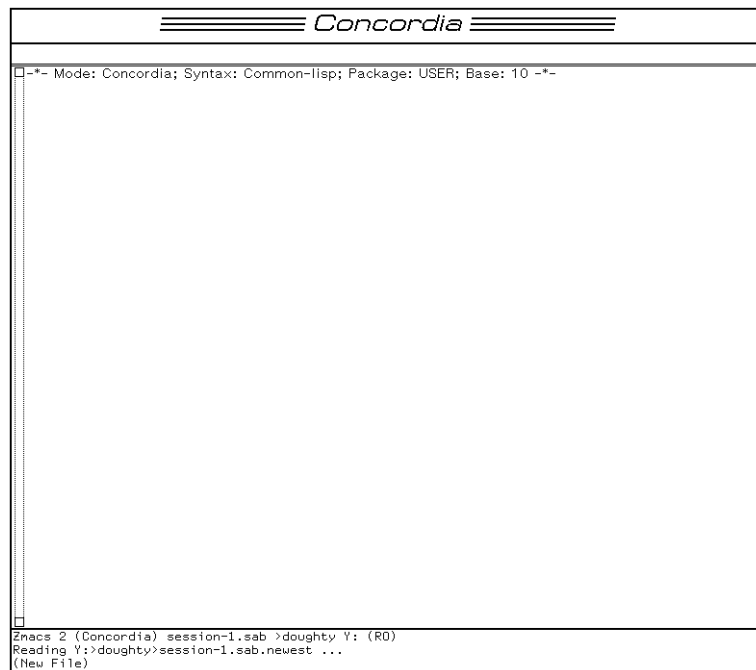


Figure 2. New .sab File with an Attribute List

### 5.3. Creating the Record

Now that you've created a .sab file, you're ready to create your first record.

#### Instructions:

1. Click Left on the Create command under **Records** in the menu.

This command asks you to supply a record name.

2. Call this new record:

Starting Up A Symbolics Computer

Press RETURN.

3. Now you are prompted for a *record type*. The default is *section*. Press RETURN to accept the default record type.
4. You are asked "What modification style?" Press L for "lock".

You just created your first record.

When you start to modify a file, you are asked for modification style. Usually you want to lock the file. The option "disconnect" is for test files where you are just experimenting and do not want to save your experiments. Each time you save a file, it is unlocked and when you next modify it, you are asked for modification style again. This prevents two writers from accidentally editing the same file at the same time.

### 5.4. Looking at the Record Template

#### Concepts:

When you create a record, Symbolics Concordia inserts a *template* for the record in your .sab file. The template has a set of *fields*, which classify and organize information about a topic. Symbolics Concordia automatically fills in the type and name of the record from information you supply; you can then go on to fill in the other fields in the record.

Let's take a look at the record you just created.

Figure 3 shows a blank record template. The fields of the record are labeled according to their functions.

At the top of the record is the word *Section*; this indicates the *type* of record. Symbolics Concordia automatically fills in the type of record you specify when you create the record.

The record name, in quotation marks, follows the record type. Symbolics Concordia automatically fills in the name you specify when you create the record. The record template includes these fields:

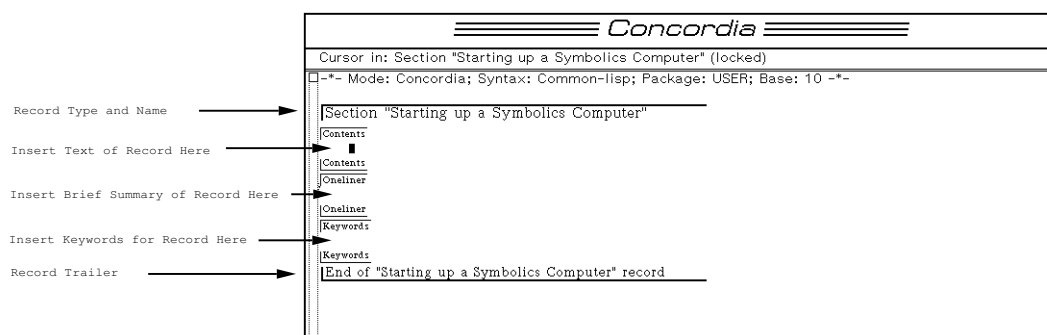


Figure 3. Record Fields

- The *Contents* field, where you type in the text of the record.
- The *Oneliner* field, an optional field where you can insert a brief summary of the contents of your record.
- The *Keywords* field, an optional field where you can insert any words or phrases that you want to appear in the index of the printed version of your documentation.

The *record trailer* follows the *Keywords* field and indicates the end of the record.

## 5.5. Putting Some Text in the Record

### Concepts:

The best way to determine how much information a record should contain is to think about the person who is going to view the record online. Since online readers can look up one record at a time, they will be most affected by how much or how little information you decide to put in each record.

For example, you may decide to put information on how to create and delete files in one record. This means that if your online readers are interested only in reading about deleting files, they have to look through a record containing information about creating files as well. This forces your readers to sift through information to find exactly what they want to read about. A better strategy is to split this information into two records, one for creating files and another for deleting files; this makes it easier for your readers to access the exact chunk of information they want.

Sometimes it may seem difficult to break up pieces of closely related information. If you find that a record relies on concepts introduced in other records, you can insert a *crossreference* to the related record or records. Crossreferences appear in the printed document or online display. For more information, see the section "Crossreference Link", page 89.

After creating two separate records for deleting and creating files, you could insert a crossreference to the record on deleting files in the record on creating files, and vice versa. Since clicking on a crossreference in Document Examiner displays the crossreferenced record, this makes it easy for the online reader to access related topics.

Another thing you must ask yourself every time you put text in the contents of a record is: *Does this record make sense when read online by itself?* Because records you create in Symbolics Concordia may be read individually, you must strive to make each record as *modular* (self-contained) as possible. We'll discuss more about creating modular records in the second session of the Symbolics Concordia Workbook, "Symbolics Concordia Workbook: Creating a Short Document".

### Instructions:

When you create a new record, Symbolics Concordia positions your cursor in the *Contents* field of the new record. This is where you write the text of a record.

Type this text in the Contents field of the record you created, "Starting Up a Symbolics Computer":

You should familiarize yourself with the operations of powering up, logging in to, and logging out of, and powering down the 3600 family of machines. Before you can do this, you should install the software on your machine and configure your site. If you are not sure that this has been done, check with your site manager. The software must be installed and the site configured before you attempt to use the system.

Figure 4 shows your record with text in its contents field.

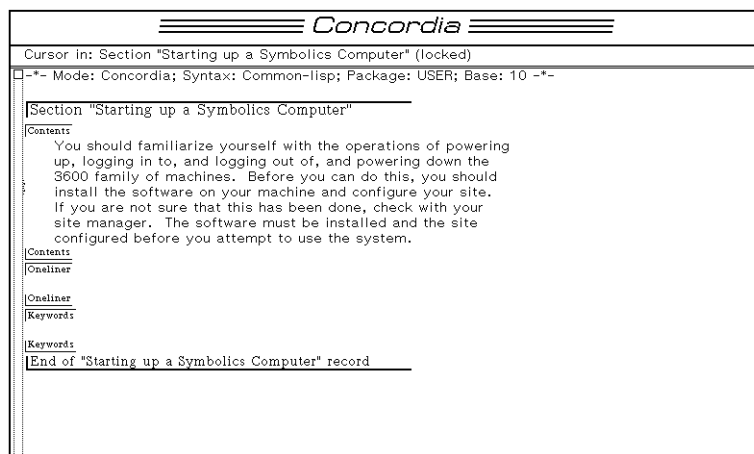


Figure 4. Record with Text

### Concepts:

Reading through the contents of this record, you see that it introduces information about starting, booting, logging in, and logging out of a Symbolics computer.

This paragraph of introductory material is in a record by itself because it is just that — an introduction. Introductory material has an important place in both on-line and printed documentation, but it need not clutter up a record that contains specific information about a single topic.

The procedures for starting, booting, logging in, and logging out all need to be in individual records, since someone might want to read only about booting a machine, or only about logging out. The goal of a record is to provide information about one topic in one place. This way, you don't slow your reader down with any information that is not relevant to the task at hand.

## 5.6. Filling in the Oneliner and Keywords Fields

You may notice two other fields in a record aside from the Contents field:

- The *Oneliner* field, where you insert a one-line synopsis of the contents of a record.
- The *Keywords* field, where you insert words or phrases that become the index entries and online lookup keys for a record.

You are not required to supply information for these two fields, but filling them in adds value to your documentation for the index and for online lookup.

To see how these work, let's fill in these fields of your new record.

### 5.6.1. Filling in the Oneliner Field

#### Concepts:

The Oneliner field of a record should contain a synopsis of the contents of a record. Even though it is called Oneliner, you can put more than one line of text in this field.

The information put in the Oneliner field is used in two ways:

1. When you use the Show Overview command for a record in Document Examiner, the information in the Oneliner field of the record displays as the summary of the record. The summary of a record helps readers to decide whether a record contains the information they are looking for.
2. The information in the Oneliner field can be used to create tables of topic descriptions using what is called the *precis view* of a record. For information on the precis view of a record: See the section "Precis Link", page 92.

Now let's fill in the Oneliner field of the record you just created.

**Instructions:**

1. Position your cursor inside the Oneliner field.

2. Type in the following text:

Introduction to the operations of powering up, logging in to,  
logging out of, and powering down the 3600 family of machines.

Figure 5 shows what a record looks like after you insert text in the Oneliner field.

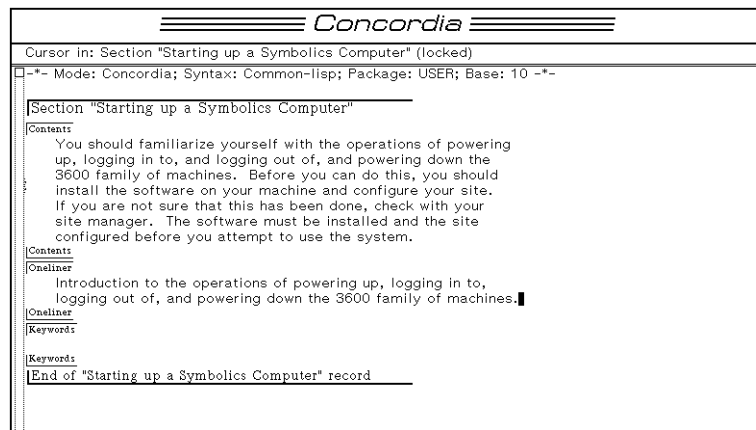


Figure 5. Record with the Oneliner Field Filled In

When you use the Show Overview command for the record *Starting Up a Symbolics Computer* in Document Examiner, the Oneliner field appears as the summary of the record.

Figure 6 shows the Oneliner field as it appears in Document Examiner.

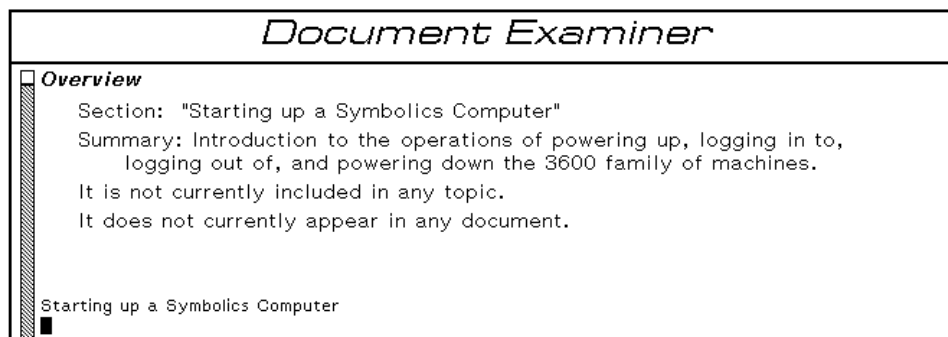


Figure 6. Oneliner Field Displayed in Document Examiner

**Workstyle Suggestion:**

Although filling in the Oneliner field is optional, in general it is a good idea to use it, since it helps online readers see what information a record contains. Try to get into the habit of filling in this field whenever you fill in the contents of a new record.



## 5.6.2. Filling in the Keywords Field

### Concepts:

Keywords have two purposes:

- In printed documentation, they become index entries.
- In Document Examiner, the Show Candidates command uses them for online lookup of documentation records.

Index entries are derived from two sources:

- Each word in the title of a record is automatically noted by Symbolics Concordia for indexing purposes.
- Any word you put into the keywords field of a record becomes a keyword, even if the word is part of a phrase.

Now let's put some keywords in the record you just created.

### Instructions:

1. Position your cursor inside the Keywords field.

2. Type the following keywords:

```
booting
logging in
logging out
```

Each entry must be on its own line. Press RETURN to end each entry.

3. Save the file using `c-x c-s`.

Figure 7 shows keywords in the Keywords field of the record. When you use the Show Overview command for the record *Starting Up a Symbolics Computer* in Document Examiner, the keywords that have been entered in the Keywords field of the record appear in the overview.

Figure 8 shows the keywords displayed in Document Examiner. Since each of the words in the title *Starting Up a Symbolics Computer* becomes an entry in the printed index, be careful not to paraphrase or duplicate the record title when you put keywords in the keywords field.

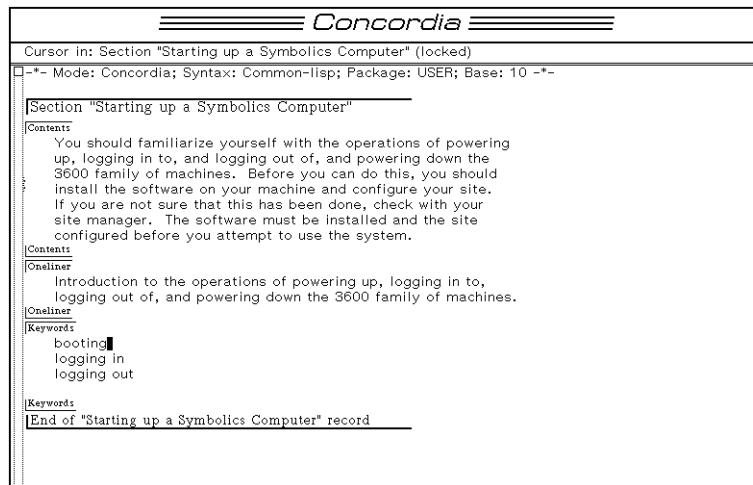


Figure 7. A Record with Its Keywords Field Filled In

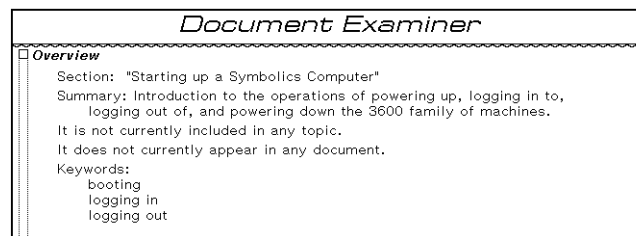


Figure 8. The Keywords Field Displayed in Document Examiner

## 5.7. Printing the Record

To print a record, press:

```
c-U s-P
```

This command prompts you for the name of a printer to use. Supply a printer name and press RETURN.

This prints the record title and the text in the Contents field.

Using `c-U s-P` is the "informal" way of printing a record, since this does not print the record with pagination. When you want to "formally" print a record, or an entire document, you must use the Page Previewer. For information, see the section "Formatting Pages Using the Page Previewer", page 202.

## 6. Symbolics Concordia Workbook: Creating a Short Document

This session shows you how to create a short document.

It assumes that you know how to create a record, as shown in the first session, "Symbolics Concordia Workbook: Creating a Record".

### 6.1. Planning Your Document

#### Concepts:

When you plan a traditional document, you begin with an outline. You plan what points you want to make, and then write text to fill in the sections of the outline.

When you plan a document in Symbolics Concordia, you begin with an outline just as you do when planning a traditional document. What's different in Symbolics Concordia is that you create records for each section of the outline.

Here is an outline for a short document that uses as part of the document the record you created in the first session, *Starting Up a Symbolics Computer*. The material in the record introduced the topics of powering up, logging in to, logging out of, and powering down a Symbolics computer.

#### A Sample Outline For Your Document

Here is the outline for this document:

##### **Starting Up a Symbolics Computer**

- I. Powering Up a Symbolics Computer
- II. Logging In to a Symbolics Computer
- III. Logging Out of a Symbolics Computer
- IV. Powering Down a Symbolics Computer

### 6.2. Creating Records for the Topics in Your Outline

#### Concepts:

In Symbolics Concordia, you can work in one of two ways:

- You can create all the records representing the sections of an outline at the same time, using a *top-down* approach.

- You can create one record at a time, filling in the contents after you create each record.

Choosing an approach is a workstyle issue; it has no effect on the final document.

An example of a task suitable for the top-down approach is the creation of a large reference manual, where you plan the outline ahead of time. After creating all the records for the manual, you can put text in the Contents field of each record as you progress in your writing. This gives you the convenience of having all of the records created before you start writing. You may wish to take the additional step of linking records together before adding contents for this type of task.

An example of a task suitable for an alternative approach is when you document bits of information at a time, without knowing how this information fits into a larger scheme. Definitions of some sort, such as functions or commands, are examples of topics for which you might use this approach. This gives you the advantage of documenting a topic and then making it part of a larger hierarchy when you know how you want to organize the material with which it belongs.

In this session we use the top-down approach, first creating all the records that will compose the document, then typing text in the contents field of each record.

#### **Instructions:**

To create the records that will compose the document, *Starting Up a Symbolics Computer*, do the following:

1. If you are not in the Symbolics Concordia Editor, select it by pressing SELECT w.
2. If the file you created in Symbolics Concordia Workbook: Creating a Record is not in the Concordia editor, read it in with `c-X c-F` and "Set Buffer Disposition" to Locked. This file contains the record, *Starting Up a Symbolics Computer*.
3. Click Left on Create under **Records** in the menu.

The Create command prompts you for a record title and a record type.

Type in the record title:

Powering Up a Symbolics Computer

Enter section for the Record type.

The new record appears below the record where you positioned your cursor.

4. Repeat this procedure to create the following records, each with the record type section.
  - a. Logging In to a Symbolics Computer

- b. Logging Out of a Symbolics Computer
- c. Powering Down a Symbolics Computer

Now you have created the records that will become your document.

**Note:** The placement of a record in a .sab file bears no relationship to where the record appears in a document, so you can arrange records within or across files for your convenience in writing and maintaining records.

The placement of records in a document is established with *links*, which will be discussed later in this session. You can, however, change the position of records in a file by using the command "Reorder Records".

### 6.3. Filling in the Contents of Your Records

Once you've created your records, you can put text in each one.

#### Instructions:

Follow these steps to type text into each record:

1. Press `␣-` to locate a record to edit.

At the prompt, type:

Powering Up a Symbolics Computer

When Symbolics Concordia locates the record, type the following text in the Contents field:

To power up and start using your Symbolics computer,  
use the following procedure:

Plug in the machine.

For the 3640, 3645, 3670, and 3675:  
Press the Power button on the front panel.

For the 3650:  
Turn the key on the front panel to PWR.

For the 3620 and 3630:  
Press the ON button on the front panel.

2. At this point you can display the record to see what the contents will look like when the record is read online or is printed. Or, you can wait until you finish filling in the contents of all the records, and then view them all at once. Use one of these methods:

- Press s-P to display the record on the screen.
  - Go to Document Examiner by pressing SELECT D. In Document Examiner use the command Show Documentation, and enter the Record title "Powering Up a Symbolics Computer".
3. Press s-. to find the next record to edit.

At the prompt, type:

```
Logging In to a Symbolics Computer
```

When Symbolics Concordia locates the record, type the following text in the Contents field:

```
Log in to the machine by typing the command Login at the
command prompt, and typing your login name. For
example, if your login name is KJones, you can log into
the default host machine, using your init file, by
typing:
```

```
Login KJones
```

4. Press s-.

When you are prompted for the name of a record, type:

```
Logging Out of a Symbolics Computer
```

When Symbolics Concordia locates the record, type the following text in the Contents field:

```
Press SELECT L to get to a Lisp Listener.
```

```
Log out by typing either the command Logout or the
function (logout).
```

5. Press s-.

At the prompt, type:

```
Powering Down a Symbolics Computer
```

When Symbolics Concordia locates the record, type the following text in the Contents field:

```
To power down your machine:
```

```
Log out by giving the command:
```

```
Logout
```

Halt the machine by giving the command:

Halt Machine

Give the shutdown command to the FEP:

Shutdown

or, press the power button on the front panel.

6. Save the file using `c-X c-S`.

Now you have created the records that will compose your document.

## 6.4. Changing the Title of a Record

### Concepts:

You cannot change the name or the appearance of a record name with conventional editor commands. To change the name of a record, use the Rename Record command. This command prompts you for a new record title.

By default, record titles follow standard capitalization rules for titles and appear in one character style. (The standard is to capitalize all words except conjunctions, articles, and prepositions that aren't the first or last word of a title.) Non-default character style and capitalization information is supplied through the Record-Name record field. This is useful if, for example, you wish to place all or part of a record name in italics.

### Instructions:

1. Press `s-`.

When you are prompted for the name of a record, type:

Powering up a Symbolics Computer

Type the command `m-X Add Record Field`. When you are prompted for a record field name type `Record-Name` and press RETURN. Type this title in the record field:

Powering up a *Symbolics* Computer

(To italicize the word *Symbolics*, place the cursor on the S and press `m-J I` RETURN.) Press `s-P` to display the record with its changed title.

Figure 9 shows this record with the Record-Name field.

Section "Powering up a Symbolics Computer"
Record-Name
Powering up a Symbolics Computer
Record-Name
Contents

Figure 9. Record with Record-Name Field Added

## 6.5. Adding Formatting Markup to Your Records

### Concepts:

The text you typed into each of the records contains no formatting commands. That is, the text contains no *formatting markup*. Formatting markup represents embedded formatting commands that make text display in certain ways, for example, as a numbered or bulleted list.

### Instructions:

This section shows you how to add formatting markup to create a bulleted list of items in the record "Powering Up a Symbolics Computer".

1. Press `s-` to locate the record "Powering Up a Symbolics Computer." Enter the name of the record. Find the following line:
 

For the 3640, 3645, 3670, and 3675:

You can imagine that this information would look better if it were displayed in a bulleted list.
2. Mark the text that begins with the following:
 

For the 3640, 3645, 3670, and 3675:

and continue marking the text until you find the last line, which is:

Press the ON button on the front panel.
3. Press `s-M`. This prompts you for a markup name. Type `itemize`, and press `RE-TURN`.

### Inspecting the Markup

1. To see what the text looks like with markup added, you can do one of the following:
  - Press `s-P` to display the record on the screen.
  - Go to Document Examiner by pressing `SELECT D`. In Document Examiner use the command Show Documentation, and enter the record title "Powering Up a Symbolics Computer".



Figure 10 shows what this record looks like when it is displayed.

#### Powering up a Symbolics Computer

To power up and start using your Symbolics computer, use the following procedure:

Plug in the machine.

- For the 3640, 3645, 3670, and 3675: Press the Power button on the front panel.
- For the 3650: Turn the key on the front panel to PWR.
- For the 3620 and 3630: Press the ON button on the front panel.

Figure 10. Displayed Record with Markup

You can add many different kinds of markup to your text. For more information on formatting markup, see the section "Controlling Your Document's Appearance", page 113.

## 6.6. Collected Record Names Pane

Each time you create a record, Symbolics Concordia places the record title in the Collected Record Names pane, which is on the bottom-right side of the Concordia editor. Figure 11 shows what the pane looks like after you create the records for your document.

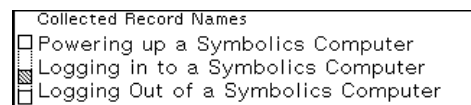


Figure 11. Collected Record Names Pane

For more information on using the Collected Record Names Pane, see the section "Using the Collected Record Names Pane", page 104.

## 6.7. Linking the Records

### Concepts:

Creating records for your document is the first part of building a document. An additional step is to use Symbolics Concordia *links* to establish relationships between records. Links are one-directional, *from* one record *to* another, target record. We say that a record containing links to other records *calls* the other records.

There are several types of links, such as the crossreference link, which are discussed in the section "Link Types". The most powerful link (because it gives a document its structure) is the *include link*. Include links make a called record behave as part of the calling record. The text of a called record is included as part of the calling record when it is displayed or printed. Using include links, you can change the structure of your document simply by changing a link, changing the order of links in a record, or by removing a link altogether.

For further information on links, see the section "Building a Document: Linking Records", page 85.

### Creating Links

The collected record names are useful when you create links. When you issue the command to create a link, you can click on one of the collected record names, and Symbolics Concordia makes a link to that record.

To create a structure for your document, you need to make include links from the record "Starting Up a Symbolics Computer" (the introduction to the document) to the other records that will compose the document.

#### Instructions:

1. Locate the record "Starting Up a Symbolics Computer", by using s-.
2. Once you find the record, position your cursor two lines after the last line of text in the record.
3. Click Left on Create Link under **Links** in the command menu. This prompts you for a documentation topic.
4. Move your cursor to the Collected Record Names pane, and click Left on "Powering Up a Symbolics Computer". Press RETURN.
5. You are prompted for a View. The default is Include. Press RETURN. This inserts an include link at the location of your cursor.

Putting an include link from the record "Starting Up a Symbolics Computer" to the record "Powering Up a Symbolics Computer" means that whenever you display the record "Starting Up a Symbolics Computer" online, or print it, the record "Powering Up a Symbolics Computer" appears as a subsection of "Starting Up a Symbolics Computer".

Now create include links for the other three records, following this same procedure.

The names of the other records, which are displayed in the Collected Record Names pane, are:

Logging In to a Symbolics Computer

Logging Out of a Symbolics Computer

### Powering Down a Symbolics Computer

Figure 12 shows the record "Starting Up a Symbolics Computer" with links.

**Note:** You can scroll the Collected Record Names pane if any of the Record titles are out of view.

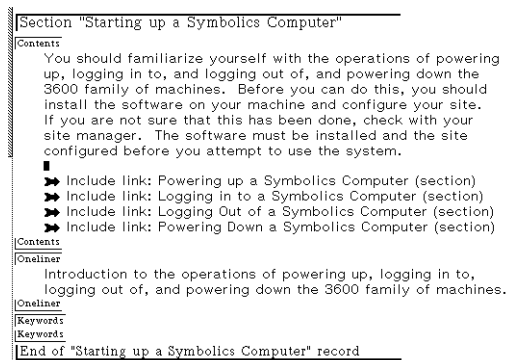


Figure 12. Record with Links

### Save the File

Save all your changes using `c-X c-S`.

## 6.8. Looking At the Document

Once you put text in records and link the records together to form a document, you will want to view the entire document to see what it looks like.

To view a document you can do one of the following:

- Display the document on the screen by pressing `s-P`. The cursor must be in the top-level record in order to display the whole document. (If `**More**`-processing is turned off, you can turn it back on by pressing `FUNCTION M`.)
- Display the document in Document Examiner, using the Show Documentation command, and entering the topic "Starting Up A Symbolics Computer".

To print the document:

1. Move your cursor to the record "Starting Up a Symbolics Computer".
2. Press `c-U s-P`. This prompts you for a printer name. Type the name of your printer.



## 7. Symbolics Concordia Workbook: Creating Graphics

This session shows you how to use the Graphic Editor, the part of Symbolics Concordia that enables you to put drawings and screen snapshots into your documentation.

### Concepts:

The Graphic Editor allows you to insert both illustrations you draw yourself, and full or partial screen images, into your document.

When viewed in a record, an image created with the Graphic Editor appears as a reference to a Graphic Editor file. In the record itself, for example, a diagram looks like this:

 Picture "gred" from file SCRC|Y:>rsw>|mdoc>tut.pic.newest

This file reference expands into a screen snapshot or a diagram when you use the Show Documentation command on the record, when you display the record, or when you print the record on a printer.

Pictures are kept in binary files. It is useful to use .pic or .dwg as a file type to identify picture file. This is suggested only as a convention; such use is not enforced by software.

To select the Graphic Editor click on the G icon at the top of the screen. (You can also press `=SELECT G.`)

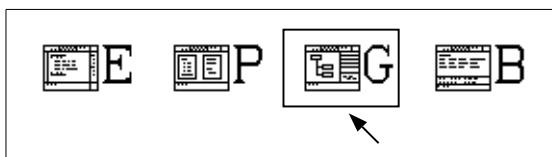


Figure 13 shows you the Graphic Editor.

For further information on the Graphic Editor, see the section "Using the Graphic Editor", page 234.

### 7.1. Creating a Diagram

Diagrams are geometric pictures you create using the Graphic Editor to insert in your documents.

#### Instructions:

To create a diagram, do the following:

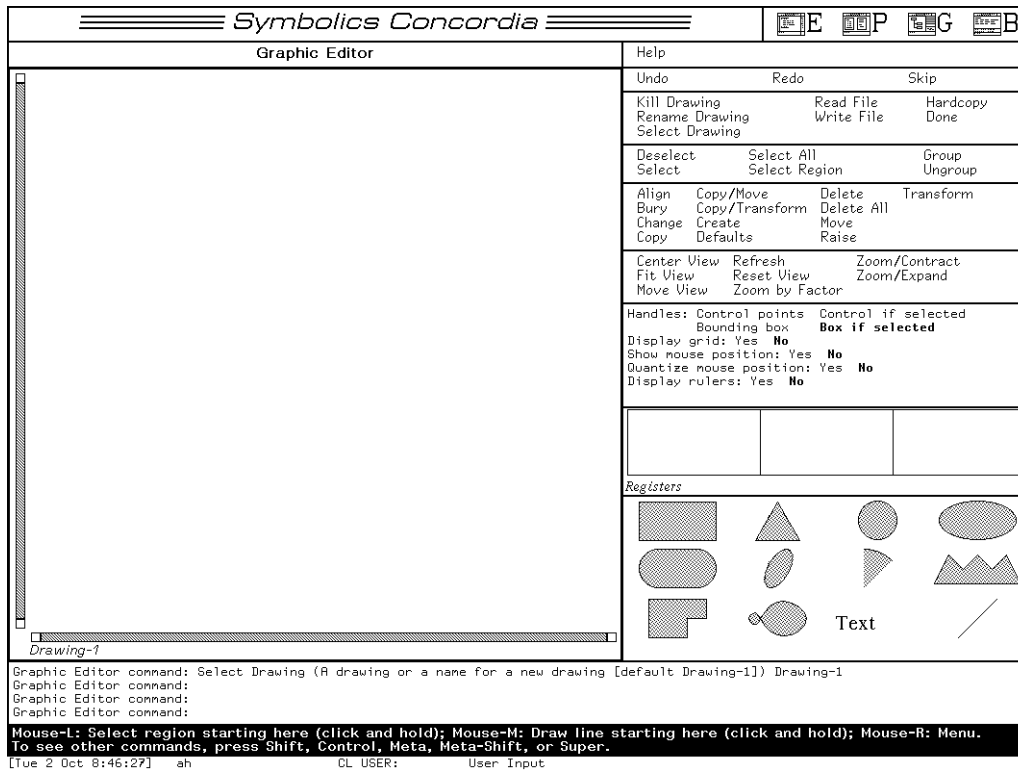



Figure 13. The Graphic Editor

1. Click on  at the top right of the screen (or press `⌘-SELECT G`) to select the Graphic Editor.
2. Move your mouse cursor to the Graphic Editor command menu, on the right-hand side of the Graphic Editor pane.
3. Click Right on **Select Drawing**. A menu appears.
4. Click Left on **New**. Below the Graphic Editor drawing pane is the Graphic Editor command prompt. You see the prompt:
 

```
Select Drawing: (A drawing or a name for a new drawing)
```
5. Enter the name of the drawing. For this example, call the drawing:
 

```
Experimental-diagram
```

 Press RETURN.

Note that the Graphic Editor does not permit spaces between words in the picture titles; you must put hyphens between words if the name of the picture contains more than one word.

6. Click Left on the circle shape in the lower right corner.
7. Move your mouse cursor to the middle of the Graphic Editor drawing pane.
8. Locate a center for the circle by clicking Left.
9. Continue to hold down the left mouse button and drag the cursor away from the center. A circle appears as you drag the cursor.
10. Release the mouse button when the diameter of the circle is about two inches.
11. Move the cursor to the Graphic Editor command menu.
12. Click Left on **Deselect**.
13. Move your cursor to **Text**, in the bottom section of the Graphic Editor command menu.
14. Click Left on **Text**. The Graphic Editor command prompt asks you for:  
Text to add:  
Type Circle.  
Press RETURN.  
Now the word Circle is attached to your cursor.
15. Move the cursor to the Graphic Editor drawing pane, to the right of the circle.
16. Click Left to place the text on the right side of the circle.

Now you can experiment with creating different shapes on your own by returning to the Graphic Editor command menu and clicking Left on a shape. Remember that after you click on the name of a shape, the mouse documentation line displays instructions on how to move the cursor to create the shape.

Figure 14 shows an example of some diagrams.

### **Save the File**

When you fill up all of the space in the drawing with different shapes, save the file.

### **Instructions:**

1. Click Left on Write File. A prompt appears, asking you for a pathname.

Here is an example of a pathname:

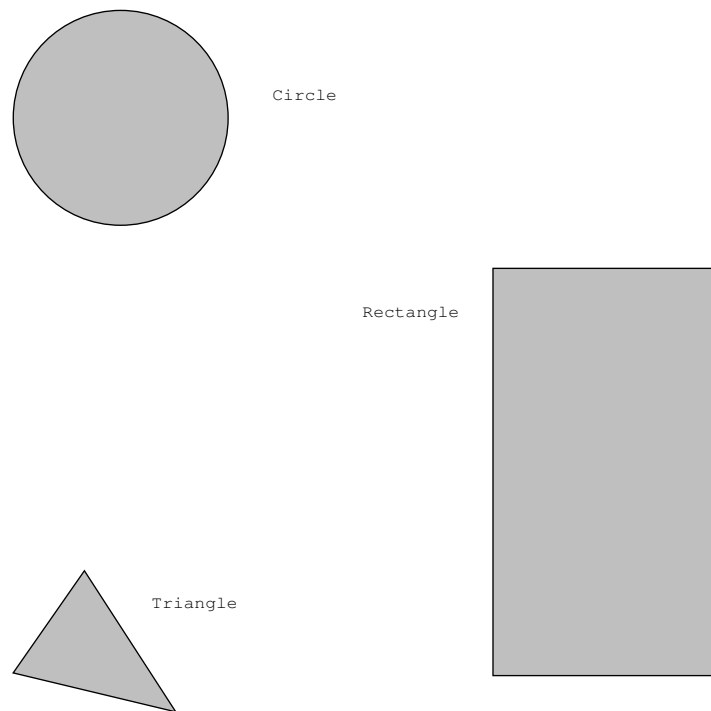


Figure 14. Example Diagrams

```
Y:>rsw>lmdoc>tutorial.pic
```

Notice that the file type is .pic.

2. Press RETURN to write the file.

## 7.2. Entering the Diagram Into a Record

Now that you've created a diagram and inserted it in a file, you can add it to a record. Return to the Symbolics Concordia editor with `␣-SELECT E`.

1. Create a record. Give it the Record title:

```
Graphic Editor Diagrams
```

Give it the type section.

2. Insert the following text into the record:

```
The Graphic Editor enables you to create different  
shaped figures and then insert them in your records.
```

```
Here is a an example of some of the shapes  
you can create:
```



3. Move your cursor two lines below the last sentence in the record.
4. Type the command `n-X Create Picture`. This prompts you for a Picture Type.

Enter:

```
Graphic-Editor
```

Press RETURN.

5. You see the prompt:

```
Enter the name of a loaded drawing or pathname
```

The default is always the name of the drawing most recently created in the Graphic Editor. In this example, the default drawing name is:

```
Experimental-diagram
```

Press RETURN.

The pathname of the drawing appears at the position of your cursor in the record. The diagram is now included in your document.

6. To adjust the appearance of the diagram, you can perform one other step:

Put the diagram in a center environment, so that the diagram will be centered on the page.

To create a center environment:

- Move your cursor to the reference to the pathname of the picture file.
- Mark as a region the reference to the picture file pathname.
- Press `s-M`. This prompts you for an environment name. Enter center.

### Display the Record

To see what the picture in this record looks like, do the following:

- Display the record using `s-P`.
- Select Document Examiner by pressing `SELECT D`, and use the Show Documentation command for the record "Graphic Editor Diagrams".

### Save the File

Save the file using `c-X c-S`.

### 7.3. Creating a Screen Snapshot

Screen snapshots are pictures of segments of a screen, or a whole screen, you create using the Graphic Editor to insert in your documents.

#### Instructions:

To create a screen snapshot, do the following:

1. Select the Symbolics Concordia Editor by pressing `SELECT W`.  
  
In this exercise you'll make a screen snapshot of the Editor commands menu, located on the right-hand side of the Symbolics Concordia editor.
2. Move your cursor to the uppermost corner of the menu, to the left of the words "Editor commands". Position the cursor so that it points to the corner of the box surrounding the menu.
3. Press `FUNCTION Q Q`. A menu pops up. Click on [Specify Rectangle] and [Named Image].
4. Click on Name: and type in a name for the image. Use hyphens instead of spaces if you want to give it a multi-word name.
5. Click on [Done].
6. A rectangle appears at the cursor location. (If the upper left corner of the rectangle isn't positioned exactly where you want it, hold down one of the `SHIFT` keys and move the mouse to relocate the rectangle.)
7. Using the mouse, drag the lower right corner of the rectangle to enclose the entire Editor commands menu.
8. Click Left to locate the lower right corner and to take a snapshot of this section of the screen.

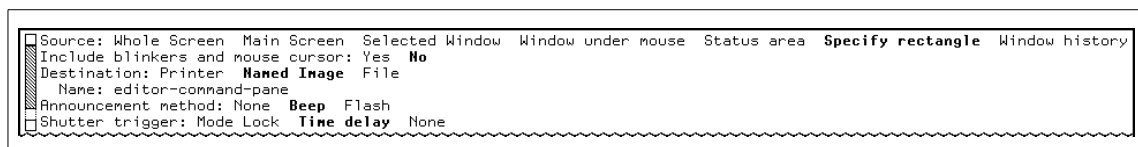
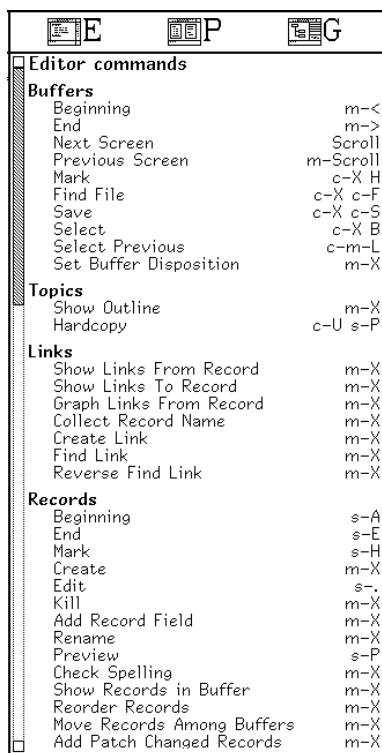


Figure 15. The `FUNCTION Q` menu

Figure 15 shows a picture of a screen snapshot of the Editor Commands pane.

**Note:** If you need to start the screen snapshot over again, you can abort the proce-




done by clicking Middle. The second time you try the snapshot, you can press FUNCTION Q directly. See the section "Hardcopying the Screen" in *Genera User's Guide*.

### Insert the Screen Snapshot in a .pic File

Once you create a screen snapshot you should insert it in a drawing so that you can use it in a .sab file.

#### Instructions:

1. Select the Graphic Editor by clicking on  in the upper right of the screen (or by pressing s-SELECT G).
2. Move your cursor to the Graphic Editor command menu, on the right-hand side of the Graphic Editor pane.
3. Click Right on **Select Drawing**. A menu appears.
4. Click Left on **New**. You see the prompt:  
Select Drawing: (A drawing or a name for a new drawing)
5. Enter the name of the drawing. In this case, call the drawing:  
Editor-commands-menu

Press RETURN.

Note that the Graphic Editor does not permit spaces between words in picture titles; you must put hyphens between words if the name of the picture contains more than one word.

6. Move your cursor to the upper-left corner of the Graphic Editor pane.

7. At the Graphic Editor command prompt type:

Add Image

Press RETURN to accept the default.

The screen image appears in the Graphic Editor drawing pane.

8. Move your cursor to position the screen image in the upper left-hand corner of the Graphic Editor drawing pane.

9. Click Left to place the image in the Graphic Editor drawing pane.

10. Move your cursor to the Graphic Editor command menu.

11. Click Left on **Deselect**.

### Save the File

You should save your drawing in a file, so that you can modify it at a later date.

1. Click Right on Write File. A menu appears, asking you for a pathname. Enter a pathname. Here is an example of a pathname:

```
Y:>rsw>lmdoc>tutorial.pic
```

2. Press END to write the file.

## 7.4. Entering the Screen Snapshot Into a Record

Now that you've created a screen snapshot and stored it in a file, you can add it to a record. Return to the editor with `␣-SELECT E`.

### Instructions:

1. Create a record entitled:

The Editor Commands Menu

Give it the type section.

2. Insert the following text into the record:

The Editor commands menu is your control panel. From here you have all the commands you need to create records for documentation. The commands are organized by category.

Here is a picture of the Editor commands menu:

3. Move your cursor two lines below the last sentence in the record.
4. Type the command `m-X` Create Picture. This prompts you for a picture type. Enter:

Graphic Editor

Press RETURN.

5. This prompts you for the name of a drawing or pathname:

Enter the name of a loaded drawing or pathname

The default is always the name of the drawing most recently created in the Graphic Editor. In this example the default drawing name is:

Editor-commands-menu

Press RETURN.

The pathname of the drawing appears at the position of your cursor in the record. The screen snapshot is now included in your document.

6. To adjust the appearance of the screen snapshot, you can perform one other step. Put the snapshot in a center environment, so that the snapshot will be centered on the page.

To create a center environment:

- a. Move your cursor to the reference to the pathname of the picture file.
- b. Mark as a region the reference to the picture file pathname.
- c. Press `s-M`. This prompts you for an environment. Enter center.

### Display the Record

To see what the screen snapshot in this record looks like, do the following:

- Display the record using `s-P`.
- Select Document Examiner by using `SELECT D` and use the Show Documentation command for the record "Editor Commands Menu".

### Save the File

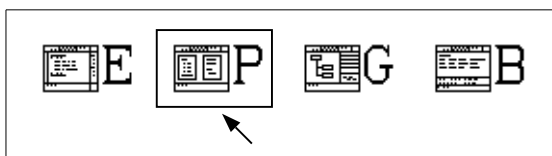
Save the file with `c-X c-S`.



## 8. Symbolics Concordia Workbook: Viewing Your Document in the Page Previewer

### Concepts:

The Page Previewer is a page layout tool that lets you preview the printed appearance of your document. Select the Page Previewer by clicking on the P icon at the top right of the Symbolics Concordia screen.



The spacing and fonts in the Page Previewer are different from those in the printed document, due to differences in resolution. The placement of words and illustrations, however, is exactly the same.

The Page Previewer also prepares documents for formal output. When you finish writing a document, if you want to examine the page layout or print it, you must process it using the Page Previewer.

The Page Previewer is useful for examining the shape of each page of your document. It gives you the opportunity to identify and fix design problems — for example, bad line or page breaks — prior to printing the document.

As you find problems in the document, you can return to the Symbolics Concordia



Editor to fix them. To return to the Concordia Editor, click Left on  at the top of the Symbolics Concordia window.

Figure 16 shows the Page Previewer.

### Instructions:

Follow these steps to view your document in the Page Previewer:

1. If it's not currently read in, use `C-X C-F` from a Symbolics Concordia window to read in the .sab file that contains the document "Symbolics Concordia Workbook: Viewing Your Document in the Page Previewer"

2. Select the Page Previewer by clicking on  in the upper right corner of the Symbolics Concordia screen.

3. Format the document by giving the following command at the Page Preview Command prompt:

```
Format Pages
```

```
Press SPACE.
```

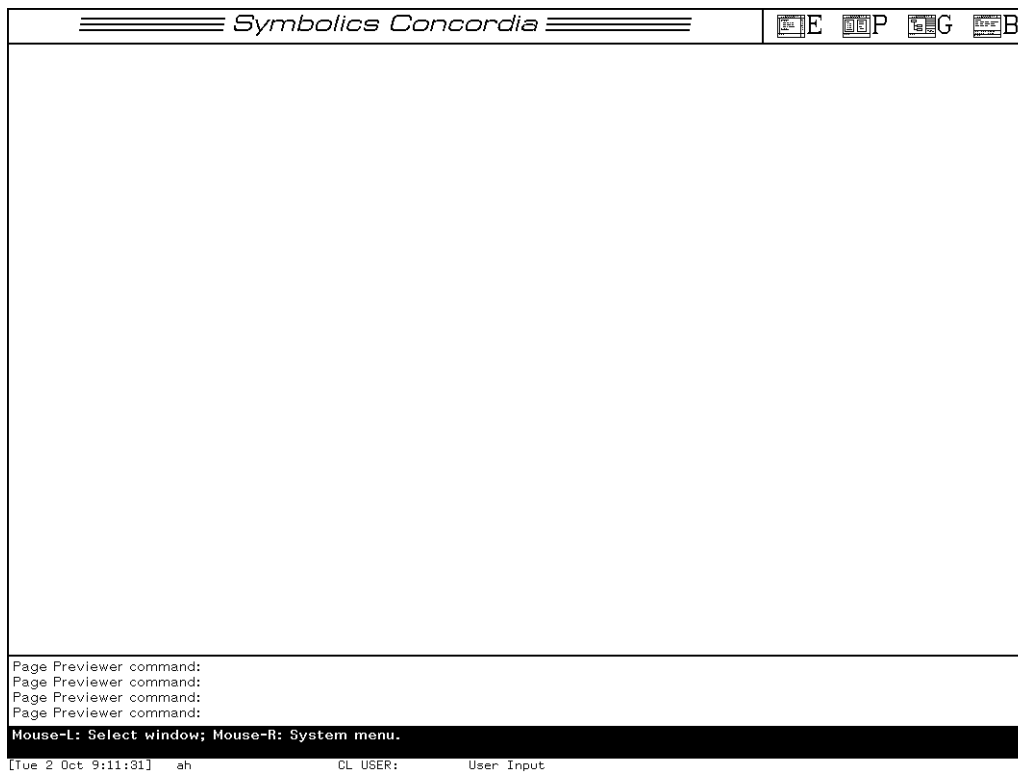


Figure 16. Page Previewer

This prompts you for a topic. Type:

```
Symbolics Concordia Workbook: Viewing Your Document in the Page
Previewer
```

Press RETURN.

After a minute or two, the command prompt returns, signifying that your document has finished formatting.

Figure 17 shows the formatted document "Viewing Your Document in the Page Previewer".

### Examine Your Document

Now that your document has been formatted, you can look at its page layout in the Page Previewer.

The first page of your document that the Previewer shows is blank, since this is the cover page. The second page of the document contains the Table of Contents. To view this, issue the command:

```
Set Page 2
```

Press RETURN.

The Page Previewer moves to the Table of Contents.



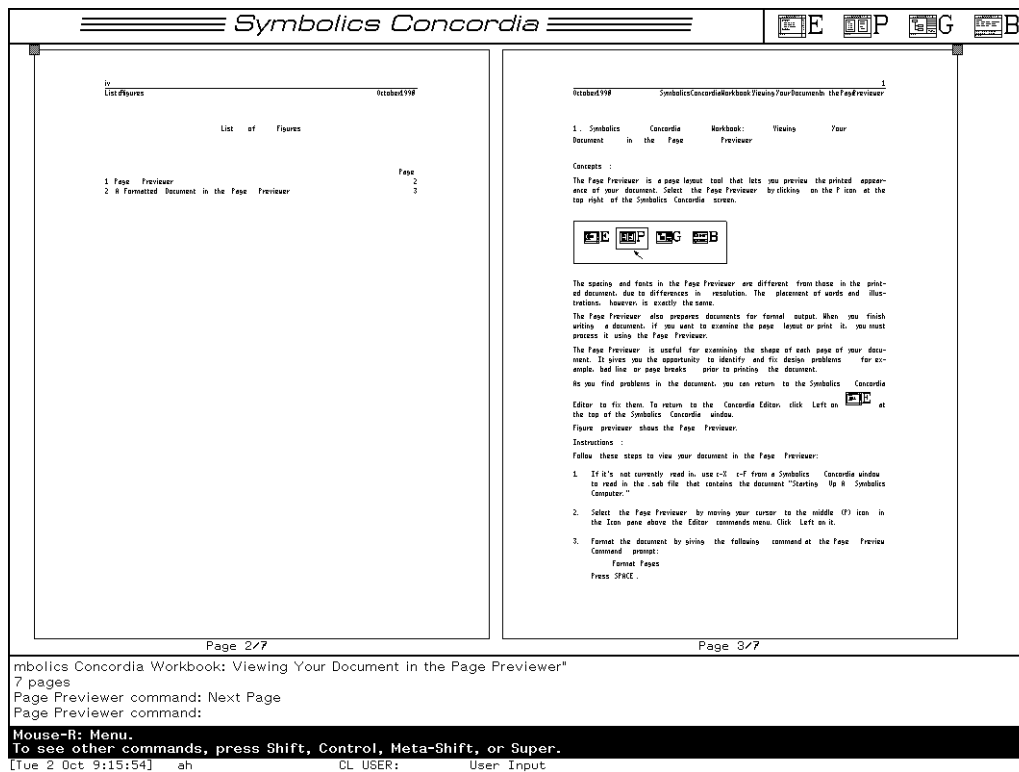


Figure 17. A Formatted Document in the Page Previewer

You can continue to look through your document, using the command:

Next Page

You can also click Left on the upper corner of the right-hand page to move you to the next page.

To go back to a previous page, specify the page number with the command:

Previous Page

You can also click Left on the upper corner of the left-hand page to move you to the previous page.

To go to a specific page, use the command:

Set Page

### Conclusion:

This section concludes the Symbolics Concordia Workbook. By following the lessons here you have learned the steps necessary to create and preview a simple document. For a more detailed description of the topics presented in the Workbook, and for a description of more sophisticated Symbolics Concordia features see:

- "Symbolics Concordia Writer's Guide"

- "Using the Graphic Editor"
- "Guide to Symbolics Concordia Book Design"

## **PART IV.**

### **SYMBOLICS CONCORDIA WRITER'S GUIDE**



## 9. Introduction to the Symbolics Concordia Writer's Guide

This document describes Symbolics Concordia, an environment for creating, maintaining, and distributing very large documentation sets with very long life cycles. It is geared to the needs of the professional technical communicator.

Its components are:

- A centralized database, which stores the documentation.
- A text editor, which combines concepts from "what-you-see-is-what-you-get" editing and generic markup languages.
- A graphic editor for creating and manipulating illustrations, which can be inserted in text.
- Configuration tools for managing document versions and distribution.
- A flexible interface for online inspection of documents during the writing and editing phase. This includes the Page Previewer and related editor commands.
- A publishing interface, which allows the documentation to be published in paper form. The formatting of printed books is separately controllable from that of the online documentation, allowing a professional book designer to customize the format.

### About this Document

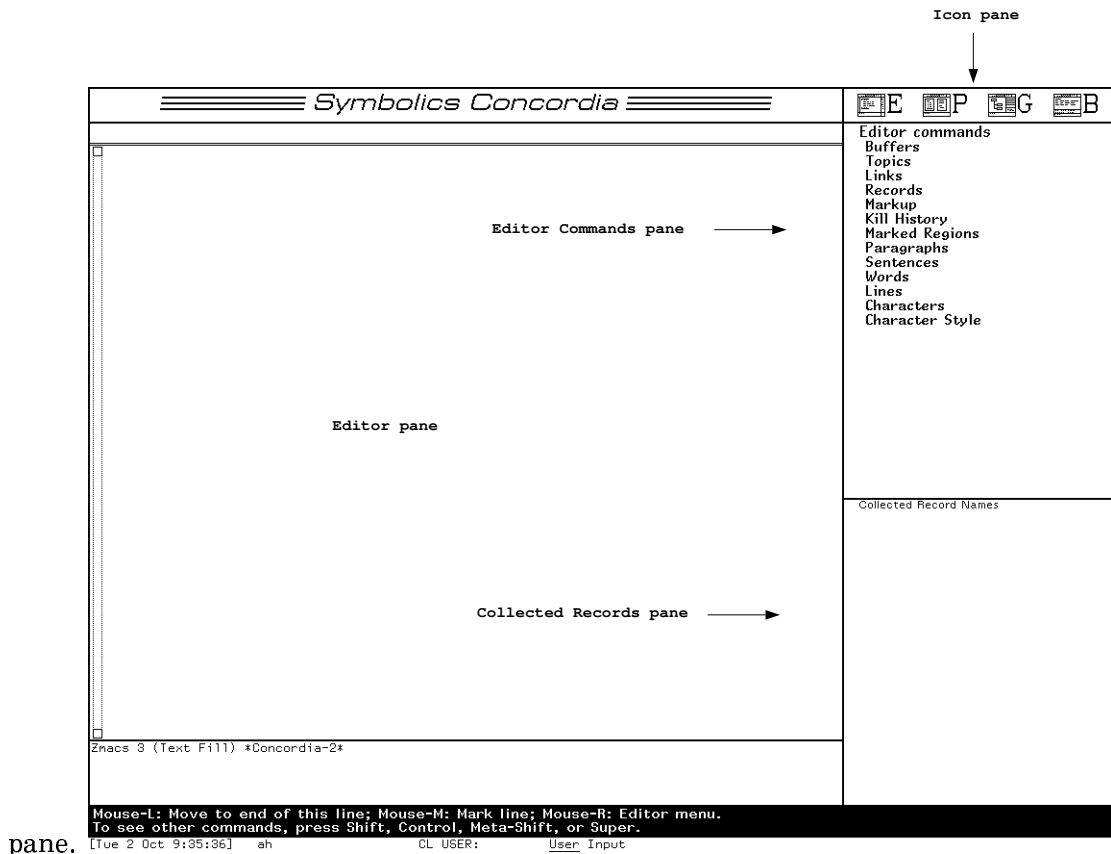
This document is aimed at professional writers and others who are familiar with text editors and formatters.

It provides task-oriented and reference information on Symbolics Concordia, especially the Concordia editor. It also presents some of the concepts and underlying methodology of document creation and maintenance with Symbolics Concordia.

Symbolics Concordia is well integrated into the Genera environment. Genera features used with Symbolics Concordia are documented in the Symbolics documentation set. Crossreferences are provided to the appropriate sections.

### 9.1. The Symbolics Concordia Window

The Symbolics Concordia window is divided into several panes: a Concordia editor pane, an Editor Commands pane, a Collected Record Names pane, and an Icon



pane.

### Editor pane

Located on the left, the Symbolics Concordia editor pane contains .sab files, the file type used with documentation records. Note: Files of any type can be read in.

### Editor Commands pane

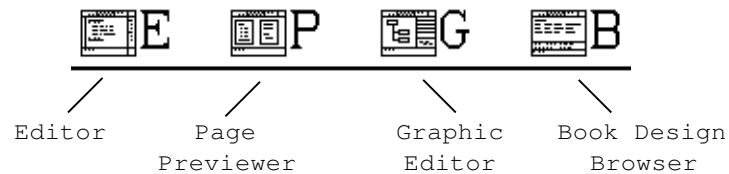
Located on the middle right, the Editor Commands pane lists by category most of the Symbolics Concordia-specific commands and some of useful Zmacs commands. All Zmacs command for fundamental text modes are available.

The editor commands menu can be customized: See the section "Customizing the Symbolics Concordia Window", page 61. Use `⌘-HELP` to see additional commands.

### Collected Record Names pane

Located on the right (bottom), the Collect Record Names pane lists the topic names of records created during this boot session as well as the names of records explicitly placed in the pane via the Collected Record Name command. The pane can be resized: See the section "Customizing the Symbolics Concordia Window", page 61. Click on a topic name when you want to create a link from one record to another or perform other operations on the record. For example, a Left mouse click activates the Show Documentation command for the record, and clicking Right presents a menu listing several other operations.

Icon pane Located on the right (top), the Icon pane displays the various subactivities available in Symbolics Concordia: the Concordia editor, the Page Previewer, the Graphic editor, and the Book Design Browser, respectively. Click on the appropriate icon when you wish to select another subactivity.



### 9.1.1. Customizing the Symbolics Concordia Window

You can customize both the Editor Commands pane and the Collected Record Names pane.

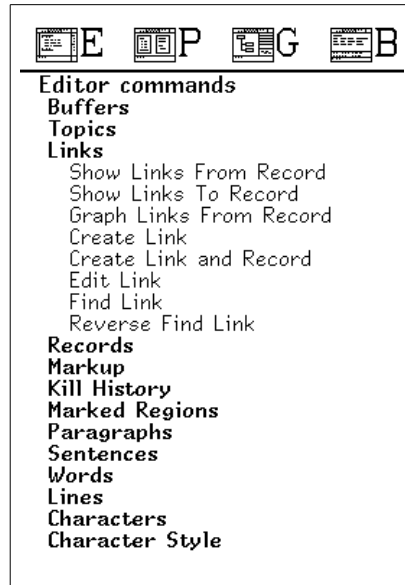
The Collect Record Names pane can be made longer or shorter. Click Left on the pane header. A bold horizontal rule appears. Slide the rule up or down by dragging it with the mouse until the pane is the desired size. Then click Left to have the change take effect.

You can alter the Editor Commands pane in several ways:

*Hiding the entries under individual command categories.*

Click Left on a boldfaced category name to hide all the entries; click Left again to redisplay the entries. This Helpful temporarily removes those categories that you do not need from the menu. It cuts down on time-wasting scrolling through the menu.

For example, when you click on the **Links** topic (for example) in the Editor command menu, it expands into the individual Link commands you can click on.



Note that this setting lasts until you boot your machine.

*Hiding the entries under all command categories.*

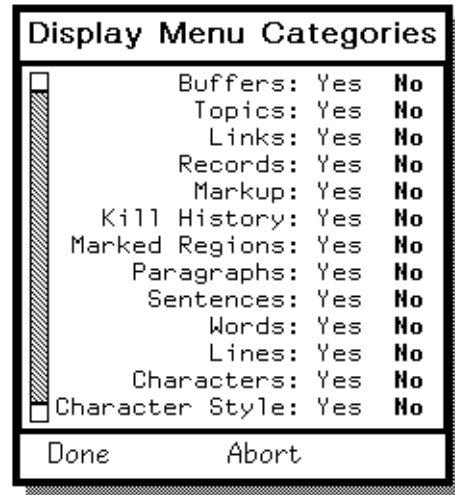
Click Left on **Editor Commands** at the top of the pane. When a menu pops up, click on Hide All. The setting is effective only during the current boot session unless you save it in your init file.

To display all the entries, click Left on **Editor Commands** again and select Show All in the menu.

*Hiding the entries under some command categories.*

The initial default is to contract all Concordia editor command menu categories. You can use `c-m-x` Insert Concordia Menu Choices to change the default expansion of editor command categories.





*Reordering the categories in the menu.*

Click Left on **Editor Commands** at the top of the pane. Then select Reorder in the menu that appears. Another menu pops up of all the categories in the thir current order. You click and hold the mouse down on a category, which highlights the name in bold. Then you drag the mouse up or down the menu to transpose the position of the highlighted name with another name. Once the categories are arranged satisfactorily, you can click on Done to have the categories in the menu reordered accordingly. Clicking on Abort keeps the categories in their original order. The customization is effective for the duration of the current boot session, unless you save it in your init file.

*Saving your customization in your init file.*

After reordering the menu or hiding one or more categories, you might want your Editor Commands menu customized automatically when you log in. Click Left on **Editor Commands** at the top of the pane. Then select Save Set in the menu that appears. This pushes a form suitable for yanking onto your kill ring. You can then yank this form, (**sage::setup-editor-menu ...**), into your lispm-init file.

*Returning to the default.*

Click Left on **Editor Commands** at the top of the pane. Click on Reset to Default in the menu.

*Refreshing the command menu.*

Click Left on **Editor Commands** at the top of the pane. Then select Redisplay from the menu.



## 10. Creating Documents with Symbolics Concordia

This section presents an overview of some of the basics of creating documents with the Symbolics Concordia editor and shows you how to begin using Symbolics Concordia.

### 10.1. Using Documentation Records

You organize documentation as a database of modules called *records*. All documents become part of one database, unless you do special programming at your site.

Each record contains a chunk of information about a specific topic; for example, this document contains a record that describes the topic "using documentation records." From these individual pieces, you eventually build an entire document, such as this one — *Symbolics Concordia*.

Each record can be retrieved from the database and read online in Document Examiner or printed on paper. See figure 18.

Records exist in Symbolics Concordia files, identified by the file type *.sab*. When you read a Symbolics Concordia file into the editor, you are working with the *buffer* associated with that file. The buffer serves as a kind of scratch pad for making changes, additions, or deletions to a set of records. These changes are made permanent and accessible to others when you write out the buffer to create a new version of the file.

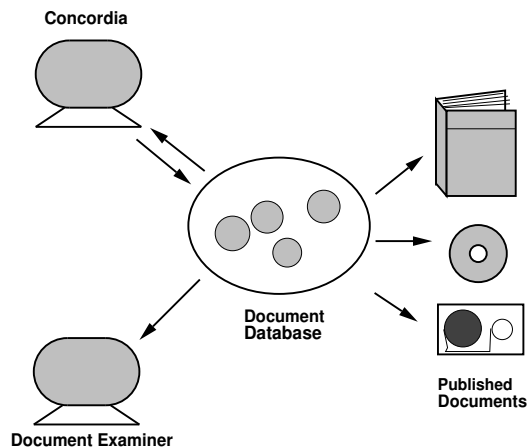


Figure 18. The authoring, printing, and viewing components all rely on the centralized database.

## Implications for Writing Style

Writers of conventional, paper-only documents can usually assume that their audience reads their books in a linear fashion, relying on the preceding sections to provide context for what follows. Since Symbolics Concordia records are individually accessible online, you cannot make the same assumption. As a result, you should carefully consider two issues when creating and editing records:

- What information deserves its own record?

A general guideline is to create a separate record for any chunk of subject matter that you think a reader is likely to need to access online through Document Examiner. Reference information and terminology definitions are easily separable into independent records; users are likely to look up documentation for a particular editor command, for example. Conceptual material, on the other hand, is harder to divide neatly.

- Does the record make sense when read online?

The process of preparing records is related to the purpose of your document and your knowledge of your audience's needs. Your readers will often view parts of your document *out of context*. This means that you must strive to make each record as *modular* (self-contained) as possible.

See the section "Workstyle Issues in Writing with Symbolics Concordia", page 211.

## Choosing a Record Name and Type

When you create a record, you are prompted to supply a *name* and a *type* for the record. Records are uniquely identified by their record name. Records are also differentiated by their type.

**Record Names.** Document Examiner uses the words in a record name to process user lookup requests. Thus, for purposes of online lookup, the name should be unique, terse, and descriptive.

Ask yourself how users might think to look up information on a particular subject, and try to use only those words that they are likely to try in their online search for information. For example, "Introduction to Symbolics Pascal" is a better name than the more general but briefer "Introduction". Another example: "Using the Mouse in the Debugger" provides more information to a user of Document Examiner than "Using the Mouse". In the case of a Lisp object, the record name should be the name of the object itself, for example, **cons**.

There are no restrictions on the length of the name. You can enter any alphabetic characters and punctuation except the em dash ( — ). The em dash must be entered specially. See the Record-Name field in "Adding Fields to a Record".

**Record Types.** Symbolics Concordia offers several record types to choose from. These record types fall into two categories:

- Object records, used in documenting language objects, such as **tv:menu**.
- Concept records, used in documenting abstract ideas that are not tied directly to code, such as "Creating Documentation Records."

Examples of object record types are Lisp function, flavor, and variable. Concept records are called sections. The documentation lookup commands use the type indicator to distinguish between like-named records. For example, if you request to see the documentation for the topic "error," Document Examiner lets you know that documentation exists for error as both a flavor and as a function.

For text records, the section type is correct for 99 percent of your records. For records documenting language objects, choose the appropriate type for the object, like function or variable. When the command prompts for a type, press HELP to see a complete list of types.

**Unique Record IDs.** Symbolics Concordia assigns each record a unique identification number at creation time. The unique id, the record name, and the record type for each record are stored in a *Record Registry*. To prevent unintentional duplication with existing records in the database, Symbolics Concordia checks the name and type you specify against the Record Registry of existing records. The registry stores such information as the record name, type, ID number, and file location. Record names for Lisp objects are also checked against the software to ensure validity. If no conflicts arise, your new record is added to the Record Registry.

**Record Groups.** There can be several versions of any record extant in your environment. Each record has a *published* version, that is the one that is installed by compilation or patching in the world. Records that you are read into an editor buffer also have an *edited* version. The Record Registry keeps track of all these versions.

## Constructing Documents

A record is the smallest unit of information accessible from Document Examiner. For readers, a record represents an independent unit of information. For writers, *a record is the basic building block used to construct entire documents.*

Writers of conventional documents impose document hierarchy by labeling portions of their books as "chapters," "sections," or "subsections." Text appears in their files in the order in which it appears in the printed books: the first chapter and its sections precede the second chapters and the sections it includes, and so on. Such a structure is fixed and relatively difficult to change.

In Symbolics Concordia, you create a document structure by making *links* between records. Links establish "relationships" among the independent records in the database. The records that you link together to form a document can span many files and can appear in any order within a file. You can easily reorganize a document by removing unwanted links and creating new ones. The order of records in a file has no bearing on the final structure of your document; the structure you build when you link records determines document structure. See the section "Building a Document: Linking Records", page 85.

## 10.2. Getting Started in Symbolics Concordia

1. First press `SELECT W` to select Symbolics Concordia. The Concordia window, shown in figure 1, is displayed.

It consists of several panes:

- Symbolics Concordia editor pane (left), which is your text editor. In the editor, you can use the special Symbolics Concordia commands, which are extensions to Zmacs (the Genera editor), as well as all the standard Zmacs commands.
  - A status line (top, left), which tells you the record the cursor is in and describes the buffer disposition. The buffer disposition can be "locked" or "disconnected." If you have locked a file buffer, no one else can write out a new version of the file. This feature is useful when several people are using one set of files.
  - Editor commands pane (right, middle), which displays a mouse-sensitive list of most of the commands available in the editor. (Press `␣-HELP` to see other commands.)
  - Collected Record Names pane (right, bottom), which lists the names of records created during this boot session as well as the names of records you explicitly place there.
  - Icon pane (right, top), which displays the icons for the three subactivities in Symbolics Concordia: the Concordia editor, the Page Previewer, and the Graphic editor, respectively. Click on an icon to move to another subactivity.
2. In Symbolics Concordia, documentation is created as modules called records, rather than simply written directly into files. Records, however, are stored in Symbolics Concordia files, so you have to create a new Symbolics Concordia file or read in an existing one before creating records.
    - Use `␣-X ␣-F` to create a new Symbolics Concordia file (or to read in an existing file).

Symbolics Concordia files must have the file type `.sab`. For example, a file containing documentation might be called `my-doc.sab`. When Symbolics Concordia prompts you for a filename, make sure to specify the `.sab` file type.

Symbolics Concordia puts an attribute list at the top of each new file:

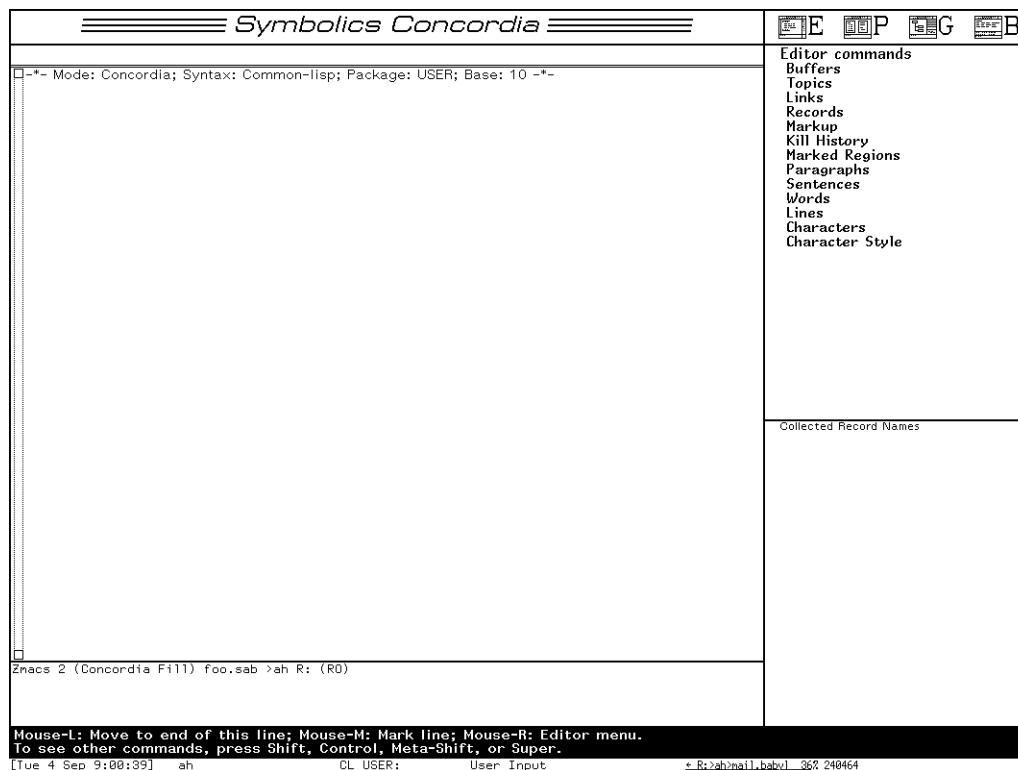


Figure 19. The Concordia window

```
-*-Mode:Concordia;Syntax:Common-lisp;Package:COMMON-LISP-USER;Base:10-*-
```

Note that the mode is Concordia. Concordia mode, in conjunction with the .sab file type, ensures that you have access to all the features provided by Symbolics Concordia.

- Use `c-X c-F` to read in a Symbolics Concordia file. When prompted, enter the name of the file, specifying the .sab file type.

Symbolics Concordia also lets you locate a specific record without having to remember which file it lives in. This feature is useful when you know the name the record you'd like to edit but can't remember the name of file it lives in.

- Use `s-.` to locate an existing documentation record. When prompted, enter the record name of your choice. Symbolics Concordia reads in the file con-

taining the record and positions the cursor at the beginning of the record.

Once you have read in or created a Symbolics Concordia file, you can begin to create records for your own documentation. For information on creating records, see the section "Creating and Filling in a Documentation Record", page 71.

### 10.3. Getting Help in Symbolics Concordia

Press `␣-HELP` to see which editor commands and keyboard accelerators are available in the Symbolics Concordia editor. See also the following sections:

- "Symbolics Concordia Reference Card", which lists the keyboard accelerators.
- "Not a Dictionary of Symbolics Concordia Commands", which lists the editor commands by task.

Press `HELP` for additional assistance when using any of the editor commands described in this section. For example, the `Create Environment` command prompts you for a markup name. To see a complete list of possible choices, press `HELP`.



## 11. Creating and Filling in a Documentation Record

You create records for each specific topic in a document; for example, this document contains a record that describes the topic "Create Record," which describes the Create Record command. You can create records in Symbolics Concordia with the Create Record and the Convert Flat Text to Record commands.

You can use these commands to create and revise records:

- "Create Record"
- "Create Link and Record"
- "Change Record Type"
- "Convert Flat Text to Record Command"

**Create Record** Adds a new record to the current buffer, prompting you for a name and a type (section, function, and so on).

When a region is marked, Create Record kills the region from the current record and yanks it into the new record. The new record appears immediately after the current record. The command creates a link in the original record to the new record, prompting you for a link type. (See the section "Link Types", page 86.)

Create Record adds the name of the new record to the Collected Record Names pane.

This command is available as Create under **Records** in the menu, or  $m-x$  Create Record.

**Create Link and Record**

Creates a record after the current record and leaves a link to it at the cursor location. It prompts for a record name, a record type, and a kind of link. (See "Link Types" for a description of link types.) If the cursor is not inside a record, a link is not created, and a warning to that effect is displayed.

If you mark a region and use this command, the region becomes the contents field of the new record.

This command is available as  $m-x$  Create Link and Record.

**Change Record Type**

Changes the record type of the record pointed to by the cursor. You are prompted for a new record type.

### Convert Flat Text To Record

Creates a new record from the marked region. The title of the record is the first non-blank text line in the region. The rest of the region is the contents of the new record. You are prompted for the record type.

The new record is placed immediately after the current one. The marked region in the current record is replaced with a link to the newly created record. It prompts for a view for the link.

When you are documenting a new topic, you can use these commands to assign a common descriptive prefix or suffix to create unique names for the new records:

- "Set Record Name Prefix Command"
- "Clear Record Name Prefix Command"
- "Set Record Name Suffix Command"
- "Clear Record Name Suffix Command"

### Set Record Name Prefix

Sets the record name prefix prepended to newly created records. Leading white space is ignored.

This command is especially useful for creating descriptive record names when you are converting flat text documents to Symbolics Concordia. See also the "Clear Record Name Prefix Command".

### Set Record Name Suffix

Sets the record name suffix appended to newly created records. Leading white space is ignored.

This command is especially useful for creating descriptive record names when you are converting flat text documents to Symbolics Concordia. See also the "Clear Record Name Suffix Command" .

### Clear Record Name Prefix

Clears the record name prefix prepended to newly created records. See also the "Set Record Name Prefix Command".

### Clear Record Name Suffix

Clears the record name suffix appended to newly created records. See also the "Set Record Name Suffix Command".

## 11.1. Symbolics Concordia Record Fields

Once a record is created, Symbolics Concordia inserts a *template* for the record in your buffer. The template contains a set of *fields*. Figure 20 shows the fields for a newly created record, which are delimited in the buffer by small iconic boxes:

- Type and Name
- Contents
- Oneliner
- Keywords

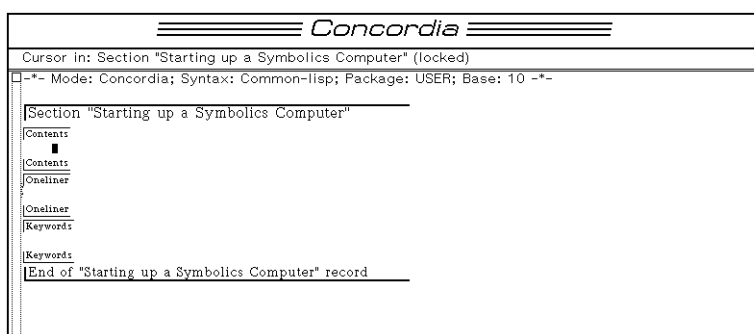


Figure 20. Concordia inserts a template for a new record.

Each field provides space for a certain kind of information. Note in the figure that a record stores far more information about a topic than just its contents. Each field has its own purpose and is used by Symbolics Concordia in various ways. One field, the Keywords field, is used in creating indexes. You can use Symbolics Concordia commands to display, print, or otherwise process various record fields. Document Examiner also uses the information in the several of the fields.

The essential fields are included in the template. The record header and trailer delimit the record and show the *type* and *name* fields. Symbolics Concordia fills in these fields from your responses to command prompts.

**Name field** Located in the header and trailer, it contains the name of the record. The name appears in upper- and lowercase and in the default character style. Once a name is entered in the field, it cannot be edited like ordinary text. To alter the name field, see "Editing the Name of a Symbolics Concordia Record".

**Type field** Located in the header and trailer, it contains the type of the record.

Record types fall into two general classes: concept records and object records. Concept records and object records are stored in different places in the record registry.

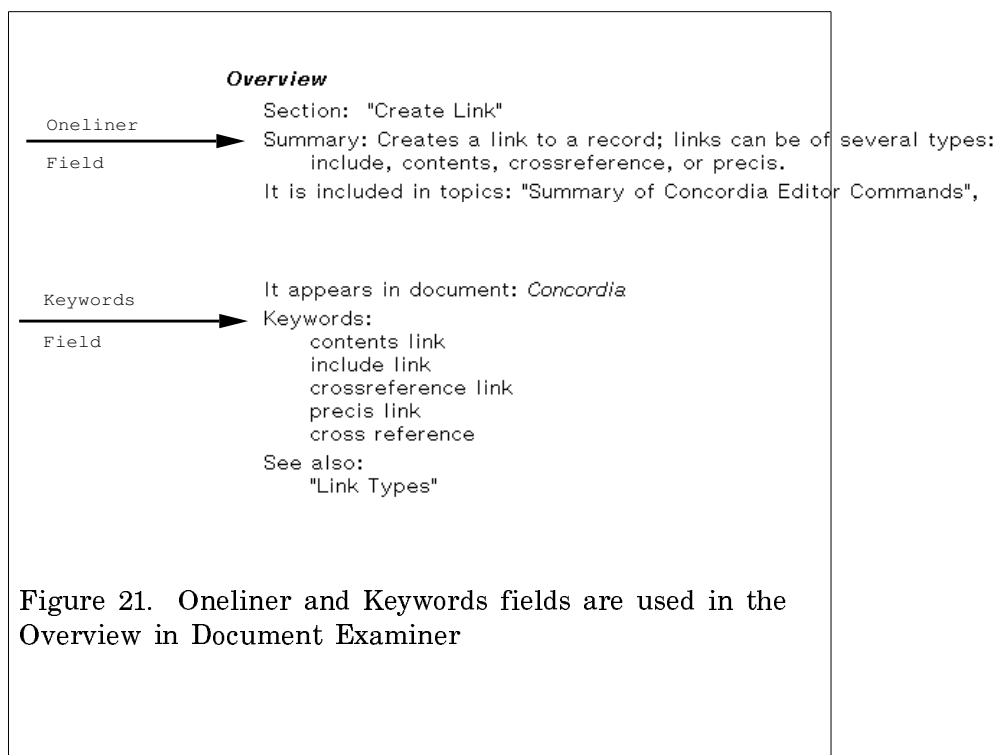
*Concept records* encompass the text records, for example, section, chapter, subsection, and appendix. It is recommended that you choose the section type for all of your text records.

*Object records* are reserved for documenting Lisp objects. There are several subclasses of object records, each of which is stored separately in the record registry. *Function* includes function, special form, and macro records. *Variable* consists of records documenting variables and constants. Records of type flavor, method, init option, message, option, property, system, resource, and meter are each stored separately.

Any two records residing in the same segment of the record registry must be given unique names.

The remaining essential fields, except for the Arglist field, are filled in by the writer.

Contents field	Main subject matter of the record.
Keywords field	Keywords (index entries) for the record, are used in several ways: <ul style="list-style-type: none"><li>• Keywords appear in the index of a printed document. See the section "Creating an Index in a Symbolics Concordia Document", page 187.</li><li>• Document Examiner finds current candidates by matching all the keywords in the database against Show Candidates requests. In this sense, your keywords become part of Document Examiner's index for automated browsing.</li><li>• Keywords are displayed in Document Examiner when you use the Show Overview command.</li></ul>

**Oneliner field**

A very brief summary of the topic. Oneliners can exceed one line of text, but should be kept as concise as possible.

Oneliners are displayed in Document Examiner when you use the Show Overview command.

Oneliners are also displayed in precis link views. See the section "Precis Link", page 92.

**Arglist field**

Included only in Lisp object records, the Arglist field contains the current arguments to the function. Symbolics Concordia automatically fills in the Arglist field from the function definition, thereby preventing typing errors.

Note: To determine whether the arguments to a function have changed since the record was created, use the `m-x Update Arglist Field` command.

**Adding Fields to a Record**

The essential fields — Contents, Keywords, and Oneliner — are automatically inserted in a new record. Symbolics Concordia provides three optional fields that you can add: Record-Name, Display-Name, and Notes.

**Record-Name**

Contains customizations for the appearance of the record name. Use this field to make any changes to capitalization, character styles, or markup for a record name. Also use this field when

you need to insert an em dash. In all other cases use the Rename Record command to change the name of a record. See the section "Editing the Name of a Symbolics Concordia Record", page 79.

Display-Name	Contains a title to be used for display, like an alias, in place of the record name. The display name is used when the record is displayed online or printed and overrides any name given in the Record Name field. Note that unlike a record name, a display name need not be unique. For more information, see the section "Changing the Way a Record Topic Displays", page 80.
Notes	Private comments or reminders. These may be notes from the document's writer or editor. The contents of the Notes field are not shown when a record is printed or displayed online.

To insert additional fields into a record, use Add Record Field.

Add Record Field Adds a field to a record; valid fields are Contents, Keywords, Notes, Oneliner, Record-Name, and Display-Name.

### **Example of a Filled-in Record**

---

Section "Looking At the Document"

Contents

Once you put text in records and link the records together to form a document, you will want to view the entire document to see what it looks like.

To view a document you can do one of the following:

Itemize

Display the document on the screen by pressing s-P. The cursor must be in the top-level record in order to display the whole document. (If more-processing is turned off, you can turn it back on by pressing FUNCTION M.)

Display the document in Document Examiner, using the Show Documentation command, and entering the topic "Starting Up A Symbolics Computer".

Itemize

To print the document:

Enumerate

Move your cursor to the record "Starting Up a Symbolics Computer".

Use c-U s-P. This prompts you for a printer name. Type the name of your printer.

Enumerate

Contents

Oneliner

Describes how to view a document online and how to print it.

Oneliner

Keywords

c-U s-P  
Printing a document

Keywords

Notes

Should we mention Format Pages and the Page Previewer here?

Notes

End of "Looking At the Document" record

---

Figure 22. A Filled-in Record





## 12. Managing Documentation Records

This section describes how to name, locate, move, and kill documentation records.

### 12.1. Editing the Name of a Symbolics Concordia Record

You can edit the name of a documentation record three different ways, depending on what you want to accomplish.

- Actually renaming the record — changing its text and punctuation with the `Rename Record` command.
- Overriding the default capitalization and character styles, using the `Record-Name` field.
- Changing the display or printing of a record name, using the `Display-Name` field.

#### 12.1.1. Renaming a Record

The record name that appears in a record's header cannot be edited like ordinary text. In order to change a record name, for example, to correct a misspelling, you must use the `Rename Record` command.

**Rename Record**      Renames the record at the cursor position. Prompts for a new name. Symbolics Concordia automatically updates the record header and trailer, all links to the record, and the online display of the record.

Use `Rename Record` to correct punctuation (except the em dash) or spelling or to change the name of a record. To change the capitalization or character styles used in a record name or to insert an em dash, add a `Record-Name` field to the record using "Add Record Field" .

If you have a renamed record and a version of that record with its old name is read into your machine (perhaps because you need to refer to an old version of a file for some reason), the record appears to revert to its old name. This can be very confusing, but it is only a local confusion in your environment. You can straighten it out by using `Rename Record` again. (It is not necessary to patch this re-naming, since it is only a local phenomenon in your environment, and does not affect any other users.) For more information, see the section "Patching a Documentation System", page 442.

**Warning:** You cannot use Rename Record to change the name of a Lisp object record. If, for example, you wish to rename because of a typing error in the record name, you must kill the existing record and create a new record with the correct name. Note that you can save any contents of the old record on the kill ring, before killing the record.

This command is available as Rename under **Records** in the menu or as `m-x` Rename Record.

### 12.1.2. Changing the Capitalization and Character Style of a Record Topic

To change a record name's capitalization or character styles, you must insert an additional record field called the Record-Name field. Use the "Add Record Field" command to add a Record-Name field. The current record name is automatically placed in the field. Change the capitalization and/or character styles as you wish. Symbolics Concordia warns you about nonstandard capitalization. To add an em dash, use the Create command under **Markup** in the menu and type Em at the prompt. (See the section "Controlling Your Document's Appearance", page 113.)

The name in the Record-Name field is an ordinary text string, so you can edit the capitalization and character styles with standard editor commands.

Symbolics Concordia updates the capitalization in the record header and trailer and in all links. Changes to the character styles are not visible in the record header in your buffer. They are visible in the record name field and when the record is displayed.

**Note:** You cannot alter the capitalization or character styles used in Lisp object records.

### 12.1.3. Changing the Way a Record Topic Displays

You might name a record "Introduction to the Page Previewer in Symbolics Concordia" to help the online reader locate the topic from among the hundreds of possible introductory topics accessible in Document Examiner. But the name would be unattractive and redundant in the printed document. Use of the Display-Name field helps solve this problem. The Display-Name field allows you to change the way a record name appears when the record is displayed or printed. Use the "Add Record Field" command to add a Display-Name field to a record. Type in the name exactly as you want it to appear, using the correct capitalization and character styles. For example, "Introduction to the Page Previewer in Symbolics Concordia" might be entered in the Display-Name field as simply "Introduction."

Remember that the Display-Name field affects *only* the display or printing of the record name. The actual name of the record appears in the record header and trailer or in the Record-Name field. You must know the actual name of a record, not the printed title, in order to locate it.

## 12.2. Locating Records

More often than not, you work on more than one record at a time. As you write text in one record, you revise the contents of another record on a related topic or add a crossreference to a record in another document.

You might also find it necessary to reorganize the text of a document as it evolves, incorporating comments from reviewers or revising information. For these reasons, you need to be able to locate your records easily.

Since record names are mouse-sensitive, clicking  $\leftarrow$  on a link — for example, a crossreference — finds that record for you. If the name of the record you want is not on the screen, you can use two other commands to find the record. Show Records in Buffer lists the names of the records in the current buffer; Edit Record finds and records any in your environment, whether or not you have read in the file in which they reside.

One useful convention is to put the name of a record that you are likely to work on again in the Collected Record Names pane. (Use Collect Record Name under **Links** in the menu.) Later, when you need to locate the record, use  $\leftarrow$ . (Edit Record) and click  $\leftarrow$  on the name in the pane. This provides a quick and accurate way to find the record you want.

### Show Records in Buffer

Displays a list of all the records in the current buffer. You can select a record by clicking on its record name.

This command is available as  $\leftarrow$  Show Records in Buffer or as Show Records in Buffer under **Records** in the menu.

### Edit Record

Finds the record you specify, reading the file into the editor buffer if the record is part of a system. It prompts for a record name; the default is the last record you located with Edit Record. You can type in a record name, or click  $\leftarrow$  on a name in the editor buffer, the Collected Record Names pane, or the Page Previewer.

Available as  $\leftarrow$ .,  $\leftarrow$  Edit Record, and as Edit under **Records** in the menu.

## 12.3. Moving Records

As you continue to create new records, you may wish to subdivide them into new files or rearrange the order in which they appear within a file. Subdividing a file with more than about forty records in it is recommended because, although there is no restriction on file size, the larger a file is, the longer it takes to read and save. This section presents information on how to move records in Symbolics Concordia.

If you are moving one or two records, you can use the commands "Remove Record From Buffer" and "Insert Record" together to move a record from one location to another (within the same file or to a different file). Use the "Move Records Among Buffers" command to subdivide a file or to move records from one buffer to one or more other buffers.

As you work on a project, you might also find that the arrangement of records has become inconvenient; the records you are currently working on are scattered throughout the buffer, and you would prefer them to be in "one place." The "Reorder Records" and "Exchange Records" commands let you rearrange the order in which records occur within a file.

#### Move Records Among Buffers

Moves records from one buffer to another. A three-column menu pops up. The right column contains the list of Symbolics Concordia buffers read into the editor. You click on two buffers to move records between them. The records in those buffers are then listed in order in the first and second columns.

You move a record name from one column to another by clicking on it with the mouse. The name appears in the same relative position in the other column. To reorder within the other column as you move it, click and hold down the mouse button; when the name appears in the other column, drag the name up or down the column. Release the mouse button when the record is positioned. When you are satisfied, you click on Done. Otherwise, terminate the command with Abort.

This command is available as  $m-x$  Move Records Among Buffers and as Move Records Among Buffers under **Records** in the menu.

**Reorder Records** Rearranges the order of records in the current buffer. A menu pops up of all records in their current order. You click and hold the mouse down on a record name. This highlights the name. Then you drag the mouse up or down the menu to transpose the position of the highlighted name with another name. Once the record names are arranged satisfactorily, you can click on Done to have the records in the buffer reordered accordingly. Clicking on Abort keeps the records in their original order.

This command is available as  $m-x$  Reorder Records and as Reorder Records under **Records** in the menu.

**Exchange Records** Reverses the order of two records. The cursor must be positioned between the records.

This command is available as  $m-x$  Exchange Records.

## 12.4. Deleting Documentation Records

There are two ways to delete documentation records. With the Kill Record command, you can permanently delete a record. *Killing* a record means removing its entry in the Record Registry as well as removing the physical record from your buffer. The Remove Record from Buffer command removes the record from the buffer, but does not permanently delete the record.

**Kill Record** Kills the specified record, removing it from the Record Registry. You are informed of any links to the record. These links must be removed before the record can be killed. After the record is killed, the record trailer and header remain with the label that this is an undocumented topic. You must patch the undocumented topic to remove it from the world. For further information on patching, see the section "Patching a Documentation System", page 442. This command is available as Kill under **Records** in the menu or as `m-X` Kill Record.

**Remove Record From Buffer**

Deletes the record pointed to by the cursor from the buffer.

Links to and from a "removed" record are not disturbed. Therefore, before removing a record, use Show Links to Record to locate and edit or delete the links.

Remove Record From Buffer does *not* remove a record from the record registry. The record is still available for insertion into another buffer with Insert Record. Use of these two commands provides a quick way to move a record from one buffer to another.

You will probably prefer to use the "Move Records Among Buffers" command if you are working with a more than a few records.

## 12.5. Viewing Records

Sometimes you might want to see only selected fields or markup in a record. For example, when you're concentrating on writing text, you might find the markup and text in the Keywords and Oneliner fields distracting. You can change your *view* of a record, that is, the fields showing in your buffer, by using the command Change Record View (`m-X`).

### Change Record View

Change Record View

Changes the view of the current record.

*Click on*                      *To display*

All fields	The entire record. This is the default.
Contents text	Text and markup diagrams in the Contents field, plus the record header.
Outline	The names of the fields and any markup diagrams in the Contents field; no text.
Record header only	The header and trailer delimiting the record; displays the record name and type.
To abort the command, move off the menu. Reinvoke the command to return to the original view.	

You can also change the record view by positioning the mouse cursor on a record header or trailer and clicking `s-sh-Middle`. A menu pops up of all the possible record views.

An alternative, Change All Record Views, offers you a choice of views for *all* the records in the buffer. You may wish to use this if you are working on only a few records. Use `s-sh-Middle` to open these records.

#### Change All Record Views

Selects the fields displayed in all the records in the current buffer. A pop-up menu of record views is displayed. After the view has been changed, the cursor is positioned at the beginning of the buffer. This command is available as `m-x Change All Record Views`.

### 13. Building a Document: Linking Records

Once you have created a number of records, you eventually want to string them together into a cohesive unit — a section or chapter, for example. In Symbolics Concordia, writers build document structure by creating *links* between records in the database. The link is the fundamental mechanism for organizing the structure of a document, from top to bottom. Very generally, *a link between two records indicates that these records have some relationship.*

Links are one-directional, *from* one record *to* another, target record. In figure 23, for example, Record C is linked to both D and E. Records that have links *to* other records can themselves be linked *from* other records. Note that Record A is linked *to* C (or said another way, C is linked *from* A). Record G, on the other hand, has no links *to* it at all, but several links *from* it. There is no restriction on the number of links made to or from a record. Record H has no links to or from other records: It is a so-called *orphan* in the database. All the other records have some relationship to one another.

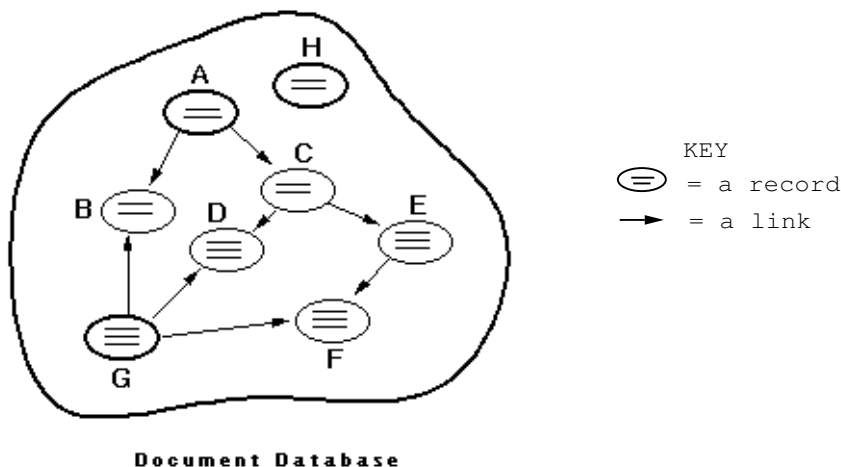


Figure 23. Links establish relationships among records.

Figure 23 indicates very generally that there are some connections among the records in the database. The exact nature of any particular relationship depends on the type of link you use to connect two records. Each link type yields its own view of a record when that record is processed by the Symbolics Concordia formatter, that is, printed or displayed online. By *view*, we mean which fields of a record the formatter selects for printing or display.

## 13.1. Link Types

How the formatter processes a link depends on the type of link you have chosen. Symbolics Concordia provides four types of links, which serve different purposes:

- *Include* and *contents* links are useful for incorporating and organizing text and building a document hierarchy.
- *Crossreference* links are useful for referring to other records by name.
- A *precis* link is useful for creating tables or brief formatted descriptions.

### 13.1.1. Using Links to Incorporate Text

#### 13.1.1.1. Include Link

The most important and frequently used link is the *include link*. The include link inserts and organizes most of the text, and sets up the structure of your document.

When a record containing one or more include links is processed, the name and contents of each "included" record are displayed or printed in the order that the links appeared in the record. Thus:

*Include link view = Name and Contents fields*

For example, figure 24 shows that the record "The Most Common Attributes" contains some introductory text and two include links. Note that the include link, marked by the arrow icon, tells you the type of link, the name of the linked record, and the type of the linked record (section).

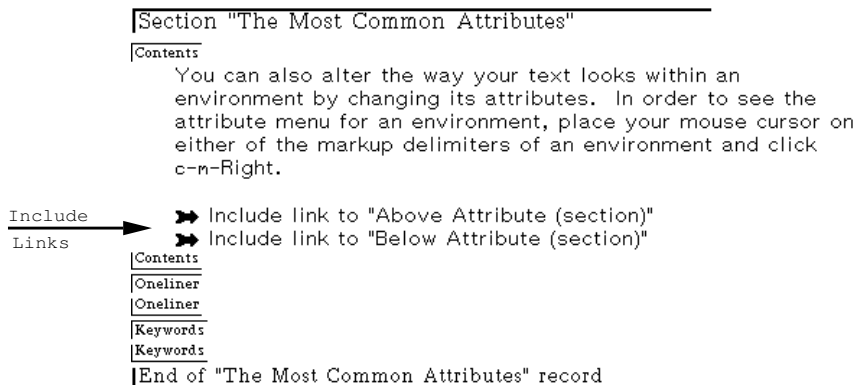


Figure 24. Include links inserted in a record.

When the record is displayed online, the formatter processes the record and its links, as shown in figure 25. The text in "The Most Common Attributes" is displayed, followed by the name of the first included record ("Above Attribute") and



its contents, and then the name of the second included record ("Below Attribute") and its contents. For a more detailed description, see the section "How the Formatter Processes a Record", page 95. Note that the record names appear in bold and on a separate line, with several line spaces above and below. The line spacing and typeface is determined by the global book design specification and cannot be altered by the writer. In cases where you wish to suppress the printing or display of a record name, use a "Contents Link" rather than an include link.

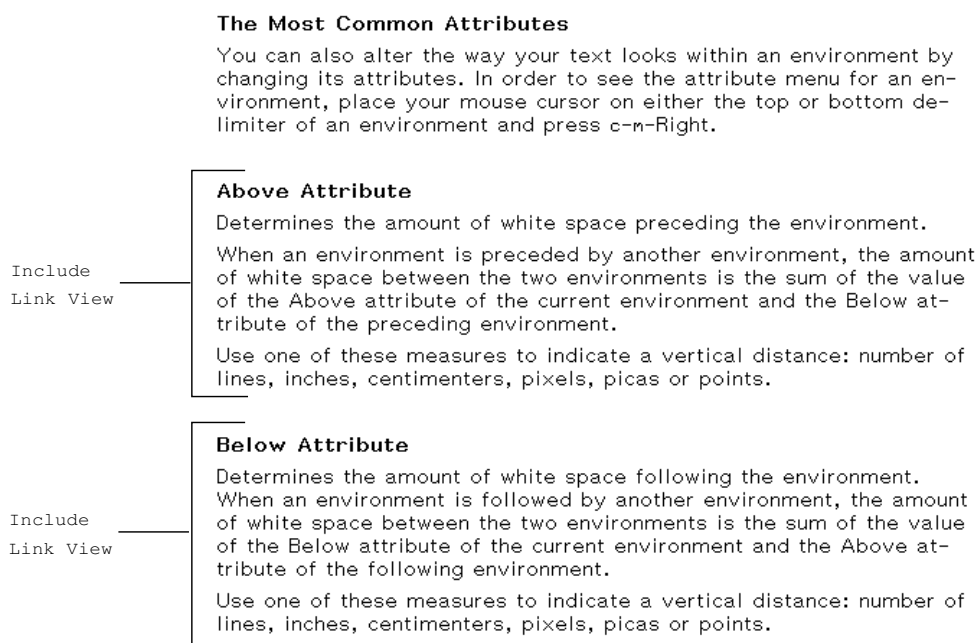


Figure 25. View produced by displaying include links.

Include links also set up the structure of your book: chapters, sections, and so on. See the section "Hierarchy of a Book", page 97.

### 13.1.1.2. Contents Link

The *contents link*, like the include link, is used to incorporate text into a document. It differs from the include link in three important ways: (1) It does not contribute to the structure of a book as shown in a book's table of contents, (2) when processed, it displays or prints only the contents of the linked record, omitting the record name, and (3) its title and keywords do not appear in the Index of the formatted document (you must use the Index command to create index entries). Thus:

*Contents link view = Contents field*

For example, the record "Above Attribute" in figure 26 contains a contents link to "Valid Vertical Distances" (also shown in the figure).

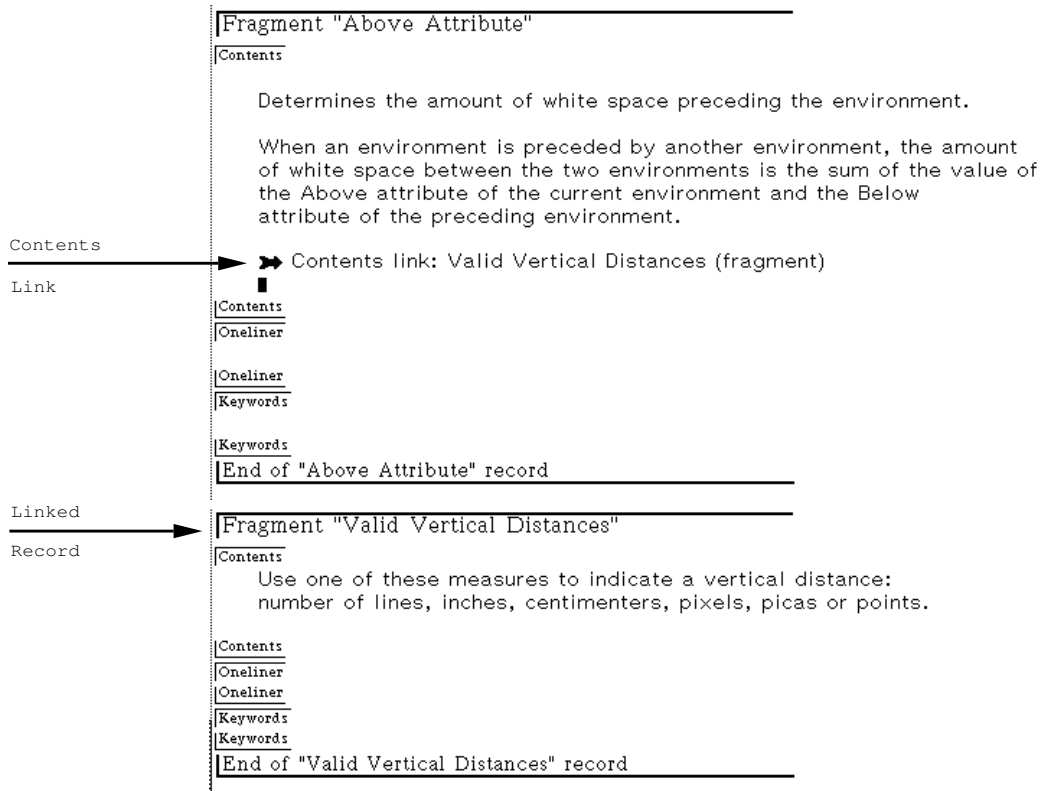


Figure 26. A contents link inserted in a record.

The processed version of the record in figure 27 shows the result: The text of "Above Attribute" is displayed followed by the one-sentence contents of "Valid Vertical Distances".

The contents link is preferable to the include link when you want to blend the contents of one record into another without displaying an intervening heading. In this example, the sentence "Use a vertical distance..." is in its own record, because this information is true for all attributes and will be used in many records. Any future modifications of this information can be made for all attributes by changing this one record.

Note, in figure 27, that the contents of the linked record are separated from the preceding text by a line break. The formatter cannot blend the processed text of a contents link into the preceding text of the linking record. Thus, you must form your paragraphs appropriately to account for the unavoidable line break.

The internal structure of the record (figure 26) is transparent to readers, who see the processed record. They have no way of knowing that this particular section, "Above Attribute", is composed of two records and is not one seamless piece of prose. As a writer, however, you must not forget that even though the record boundaries are "invisible" in the current context, the linked record is still individually accessible to readers in Document Examiner. When this record, used here as

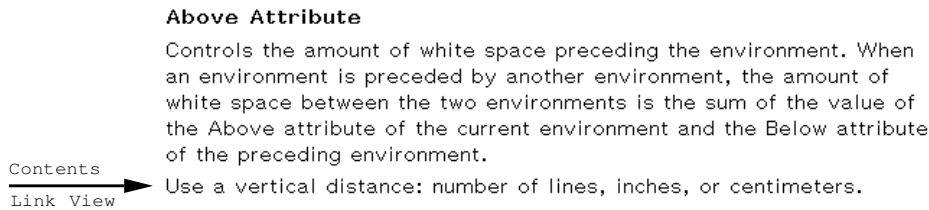


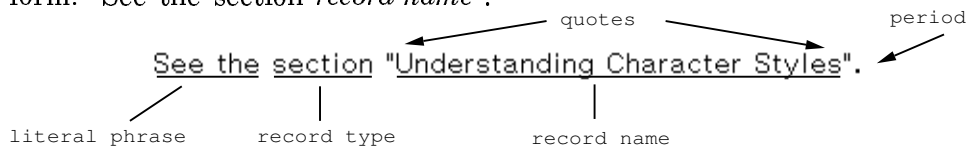
Figure 27. View produced by displaying a contents link.

the target of a link, is displayed on its own, its name and contents are displayed as usual.

## 13.1.2. Using Links to Refer to Other Records

### 13.1.2.1. Crossreference Link

A *crossreference link* allows you to refer to other records in the database. There are several forms of a crossreference link. By default, a crossreference is of the form: "See the section *record-name*".



Other options are: a *topic* view, which presents only the record name, or an *invisible* view, which allows you to make arbitrary text mouse-sensitive by associating that text with an unseen crossreference form.

Figure 28 shows some crossreference links.

When the record is displayed online, as shown in Figure 29, you see a full sentence. Thus:

*Crossreference link view = Type and Name fields*

A crossreference link offers major advantages over a typed crossreference.

- Crossreferences are automatically updated when you edit the name of the referred-to record.
- Crossreferences are mouse-sensitive in a Document Examiner viewer, as shown in Figure 29. When a reader clicks on one, Document Examiner displays its associated documentation.
- Crossreferences are displayed in Document Examiner when a reader invokes the Show Overview command. They are listed under "See also" in the overview and are mouse-sensitive.

Note that online, crossreferences #1 and #2 are formatted identically; both print the words "see the", the record type, and the record name in quotations. (See Figure 29.)

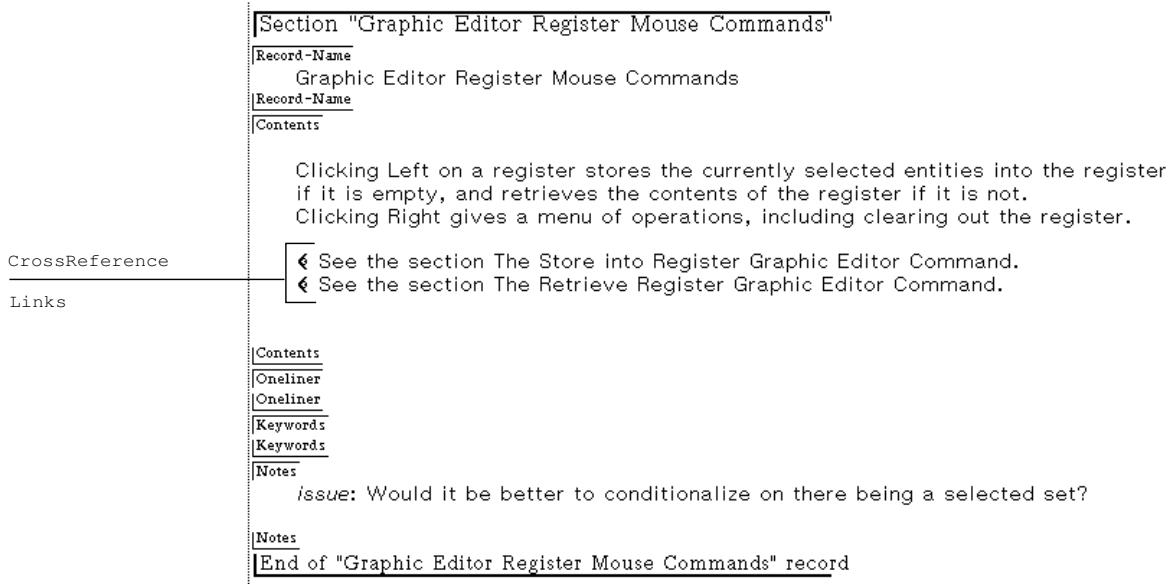


Figure 28. Crossreference links inserted in a record.

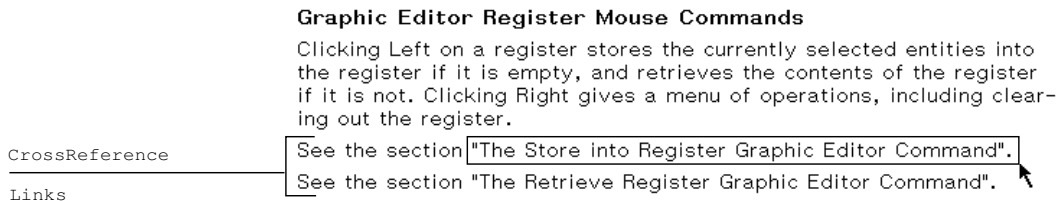


Figure 29. View produced by displaying crossreference links.

You have no way of knowing whether the linked record is included (in the sense of an include link) in the same book as the linking record, or not. And there's no reason you need to know this information in an online reference, since (1) you are looking at an individual record, and not a section within the context of a book, and (2) the crossreference is mouse-sensitive; you can locate the documentation simply by clicking on the reference.

When the record is printed, as opposed to displayed online, the crossreference provides additional information necessary for locating the documentation easily:

- If the linked record and the linking record are included in the same book, the formatter prints ", page" and the page number on which the linked record appears. For example, in print, crossreference #2 looks as follows:

menu and type Em. (See the section "Controlling Your Document's Appearance", page 39.)

- If the linked record and the linking record are *not* included in the same book, the formatter prints "in" and the title of the book including the linked record. For example, in print, crossreference #1 looks as follows:

output device. See the section "Understanding Character Styles" in *Genera User's Guide*. To change the appearance of a record, name...

### Using the Correct Spacing for Crossreference Links

You must take care with the spacing around crossreference links. As is true for all types of links, the markup appears on its own line in a record. However, when the formatter processes the link, it embeds the crossreference in the surrounding text — *without any character spacing*. To get one character space preceding and following the crossreference, you must explicitly press the SPACE bar before and after creating the link. For example, note the space in the line following crossreference link #1 in figure 28. The space preceding the link is there too, but is not visible in the record. Figure displayed-crossref-links shows the correct character spacing would be left around the displayed crossreference. To omit spacing, close up the text surrounding the link as much as possible. Note the placement of each parenthesis in crossreference link #2 (figure 28) to produce crossreference #2 within parentheses (figure displayed-crossref-links).

See the section "Run-in Markers", page 114 for further information on markups and spacing.

### Topic Crossreferences

The default crossreference link produces a standard crossreference. If you need a more flexible style of crossreference, you can modify the crossreference link to produce a topic crossreference. This variant produces a link that omits the introductory "See also" phrase and displays only the referenced record's topic. The tradeoff is a loss of accuracy in the printed reference; a topic-style reference never prints a page number or book title. When you feel this information is important for users (for example, when the reference is separated from the referred-to section by a great many pages), use a standard cross reference link instead.

### Invisible-style Crossreferences

Use the *invisible* view of a crossreference in conjunction with the **CrossRef** environment to associate text with a crossreference to another record.

The **CrossRef** environment causes the text contained inside it to be made mouse-sensitive as a crossreference to the first link contained inside the environment.

For example,

```

|CrossRef
  Transform
  ↪ Invisible link to "The Transform Entity Graphic Editor Command" Section
|CrossRef
  and
|CrossRef
  Copy/Transform
  ↪ Invisible link to "The Copy and Transform Entity Graphic Editor Command" Section
|CrossRef
  allow you to do two-dimensional transformations on one or more entities.

```

The resulting crossreferences appear in running text as:

Transform and Copy/Transform allow you to do two-dimensional transformations on one or more entities.

The words "Transform" and "Copy/Transform" will be mouse-sensitive online, and clicking on them will display the documentation in the indicated records.

### 13.1.3. Using Links to Create Tables

#### 13.1.3.1. Precis Link

The *precis link* is useful for creating dictionary-like lists or definition tables so commonly used in technical manuals. When a record containing one or more precis links is processed, the name and oneliner summary of each linked record are displayed or printed exactly where the link appeared in the record. The name of the record is mouse-sensitive. In addition, if the linked record is a Lisp object record (for example, a record of type function), the argument list is displayed. Thus:

*Precis link view = Name, Arglist (if a Lisp object record), and Oneliner fields*

For example, "Summary of the Most Common Attributes", in figure 30, contains precis links to two section records, also shown in the figure.

When the record is processed (see figure 31), the name of each linked record appears at the left margin followed on the next line by its oneliner summary. The line spacing above, below, and between the displayed records is under the control of the global book design specification.

You might not be satisfied with the default display. To change some aspects of its appearance, embed the links in one of the many formatting environments provided in Symbolics Concordia. (For background information on environments, see the section "Controlling Your Document's Appearance", page 113.). For example, enclosing several precis links in "Description Environment" markup (shown in Figure 32) renders the processed links in a two-column format. See Figure 33.

Precis links are particularly handy for listing Lisp function records in a table format. Figure 34 shows a table of Lisp functions, created by embedding three precis links in a Description environment. Note that the function names and argument lists are rendered in the conventional typefaces used by Symbolics documentation: bold and italics, respectively.

The previous examples have illustrated the use of precis links for Lisp function

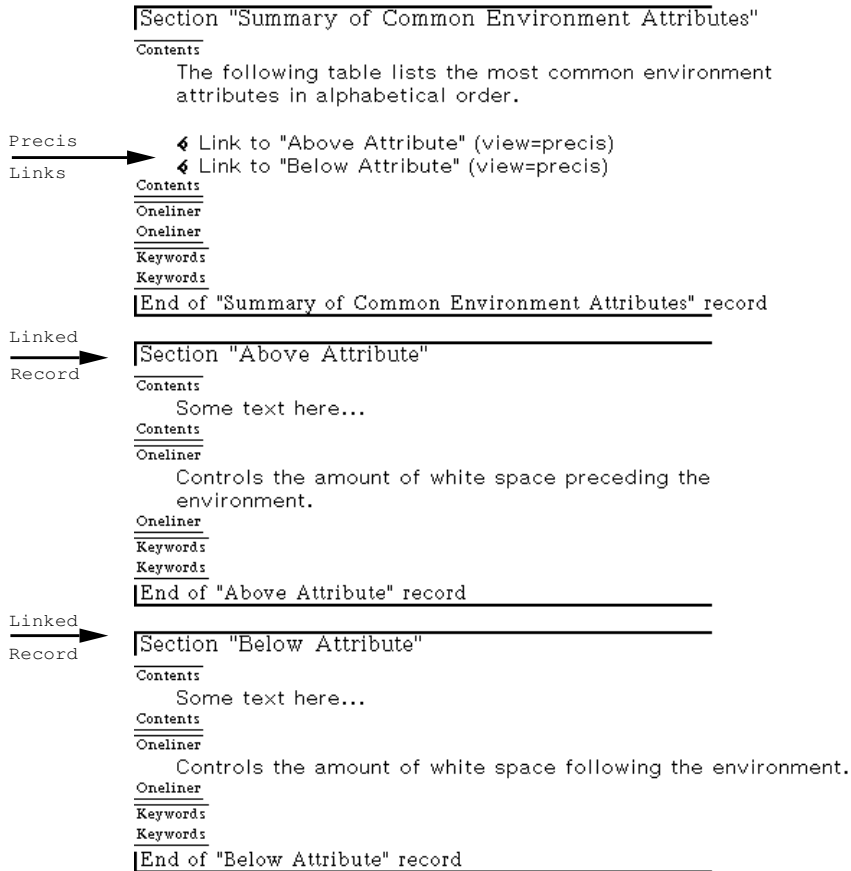


Figure 30. Precis links inserted in a record.

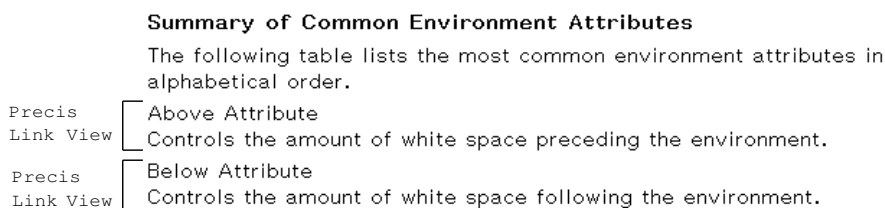


Figure 31. View produced by displaying precis links.

records and records documenting commands or terminology. In addition, precis links can be used to create a mouse-sensitive "table of contents" within a record. For example, figure 35 is excerpted from an introduction record that lays out the topics to be covered in the rest of the section. Note the table of contents is organized as a bulleted list, indicating that the writer has wrapped the precis links in an Itemize environment. Note also that each name is mouse-sensitive, as indicated by the highlighting box.

Precis links offer two major advantages over manually prepared tables and lists.

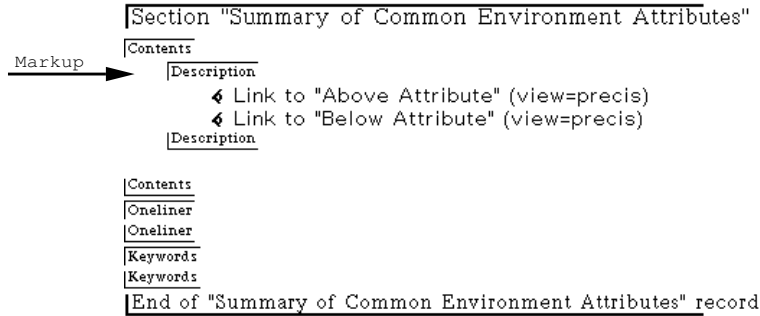


Figure 32. Precis links embedded in a Description environment.

### Summary of Common Environment Attributes

Above Attribute Controls the amount of white space preceding the environment.

Below Attribute Controls the amount of white space following the environment.

Figure 33. A Description environment changes the appearance of the precis link view.

### Functions that Find the Home Package of a Symbol

**symbol-package** *symbol*  
Return the package in which *symbol* resides.

**sys:package-cell-location** *symbol*  
Return a locative pointer to *symbol*'s package cell.

**keywordp** *object*  
Check if *object* is a symbol in the keyword package.

Figure 34. Precis links display the argument list of Lisp functions.

- Updating is much easier. When you change the Name, Arglist, or Oneliner field in the linked record, the precis link has access to the updated information. For function records, in particular, this saves much typing corrected Arglist fields.
- The name is mouse-sensitive, so that readers can quickly get to the full documentation for the topic. The writer gives the reader the freedom to directly pursue any topic of immediate interest.



### Everything You Ever Wanted to Know About Links

This section describes the following information. Click on any of the topics to see its documentation.

- [Link Types](#)  
Describes the five different types of links.
- The Significance of Links  
Describes how links affect the structure of your document, its accessibility to online readers, and your writing style.
- Using the Collected Record Names Pane  
Describes how to use the Collected Record Names pane to create a link.
- Link Commands  
Describes the different commands that insert links in a record.

Figure 35. Precis links create a mouse-sensitive table of contents.

## 13.2. The Significance of Links

Links, particularly include and contents links, are the only means Symbolics Concordia provides to incorporate and order text in a document. In addition, the way in which you arrange your links has significant impact on the structure of your document, its accessibility to online readers, and, not least of all, your writing style. Decisions about which records should get linked to which other records should be made thoughtfully, since:

- Links create the hierarchy in a book: chapters, sections, and so on.
- Links create a reading path through a topic for Document Examiner users.
- Links, which support Symbolics Concordia's record-based documentation system, enforce a modular writing style. See the section "Workstyle Issues in Writing with Symbolics Concordia", page 211.

To really understand these issues, you first need to understand how the Symbolics Concordia formatter processes a record, in particular its links.

### 13.2.1. How the Formatter Processes a Record

When you display or print a record, the formatter processes the links as they appear in the record, following the trail of links down to the end of the tree before processing more text or the next link branch.

For example, figure 36 shows that "The Most Common Attributes" (upper left) contains a link to "Above Attribute" and then a link to "Below Attribute". In addition, both "Above Attribute" and "Below Attribute" are linked to "Valid Vertical Distances". Accordingly, the formatter processes "The Most Common Attributes" as follows:

1. Initial text in "The Most Common Attributes"
2. Link to "Above Attribute"
3. Link to "Valid Vertical Distances"
4. More text in "The Most Common Attributes"
5. Link to "Below Attribute"
6. Link to "Valid Vertical Distances"

Note that

- Because the formatter processes links in the order in which they appear within the record, it processes the link to "Above Attribute" (step 2) before the link to "Below Attribute" (step 5).
- Because the formatter processes links in top-down order, it processes the link to "Valid Vertical Distances" (step 3) before it processes the link to "Below Attribute" (step 5).

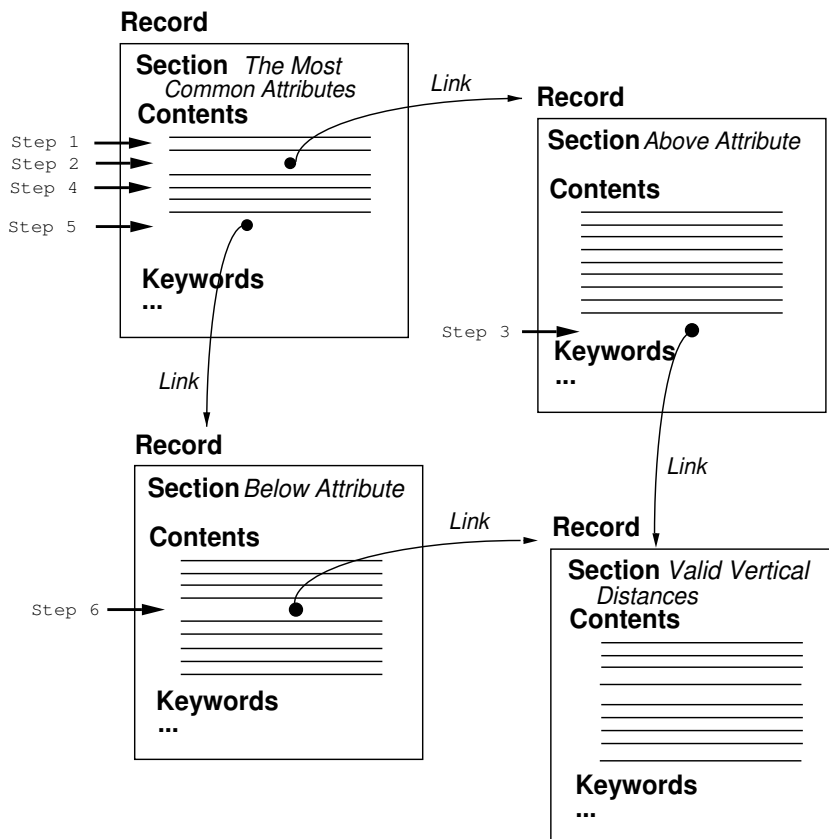


Figure 36. How the formatter processes links.

### 13.2.2. Hierarchy of a Book

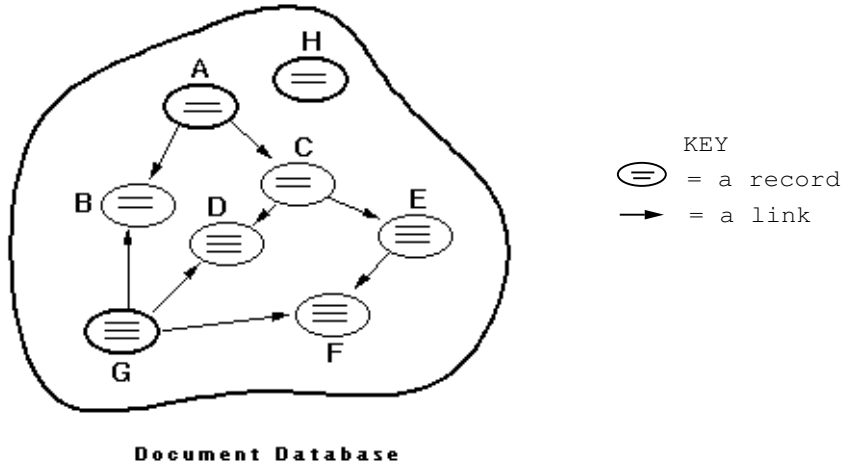
For most readers, the table of contents (figure 37) lays out the structure of a book. The typeface and size, indentation, and numbering all provide clues to how the book is organized, for example, which unit of text are chapters, sections, or subsections; which sections are subsumed by which chapter, and which subsections fall under which sections.

USER'S GUIDE TO SYMBOLICS COMPUTERS	
Table of Contents	
<b>1. Overview of Symbolics Computers</b>	<b>1</b>
1.1 Documentation Notation Conventions	1
1.1.1 Notation Conventions Quick Reference	1
1.2 Introduction to Genera	3
1.2.1 The Console	3
1.2.2 The Screen	4
1.2.3 The Display	4
1.2.4 The Keyboard	4
<b>2. Starting up</b>	<b>15</b>
2.1 Powering up	15
2.2 Logging in	17
2.3 Logging out	17
2.4 Powering down	18

Figure 37. The table of contents shows the structure of a book.

To produce a structure like the one presented in the figure, a writer using a conventional documentation system probably would have had to explicitly label or number certain chunks of text as chapters and others as sections, and so on. In Symbolics Concordia, include links create an equivalent book structure. How a record is "include-linked" to and from other records solely determines whether it will become a chapter, section, or other unit in the finished book. (Note: The other link types, while they contribute to the complexity of your document, do not affect its *visible* structure, as shown in a table of contents.)

The following picture of the documentation database helps illustrate very generally how the Symbolics Concordia formatter creates a book structure.



Example: Record A is processed by the formatter. Assume that all links are include links and that the links were inserted in alphabetical order. According to the rules of formatting, described in "How the Formatter Processes a Record", when we print Record A, the links are processed as follows:

- First A prints B and C
- Then C prints D and E
- Finally, E prints F

In outline form, the hierarchy is:

```

A
  B
  C
    D
    E
      F
  
```

*The include link, in effect, instructs the formatter to make the linked record a subdivision of the linking record.*

Now assume that Record A is a book; that is, when the formatter processes Record A, the entire book is printed. The hierarchy thus produced corresponds to the standard book structure, as shown in the table.

<i>Record</i>	<i>Corresponds to a ...</i>
<b>A</b>	Book
<b>B</b>	Chapter
<b>C</b>	Chapter
<b>D</b>	Section
<b>E</b>	Section
<b>F</b>	Subsection

Note that records linked from the same record are at the same level in the hierarchy; for example, D and E, both sections, are linked from C. Also note that a record's position in the overall hierarchy is determined by its relative position in the linking arrangement.

### 13.2.2.1. Overall Book Structure

From the Symbolics Concordia writer's point of view, the structure of a book is built from all the links in the document, not just the include links, which represent the visible structure of a book. Each link, no matter what its type, contributes to the finished product, whether or not the linked record appears in the table of contents. For example, contents links, when processed, do not appear in the visible book structure, since the name of the linked record is not printed and the text is blended into that of the linking record. Yet, conceptually the contents link is certainly part of the book's structure. Make sure you examine the overall structure of your book with care. See the section "Examining the Organization of Your Document", page 109.

### 13.2.2.2. Setting up the Hierarchy of a Book

Include links from record to record tell the formatter which records are generic subdivisions of other records. To tell the formatter to specifically label certain subdivisions as chapters, sections, and so on, requires a special mechanism called a script record.

The *script record*, literally a record of type script, defines one or more records as a book. Like any record, the script record can contain text and illustrations, and formatting markup. Unlike any other record, however, it sets out the book design specifications and produces the standard book hierarchy, from top to bottom. This discussion focuses in very broad terms on the script record's role in organizing a book.

The script record is informally known as the *top-level record*, and for good reason. When the script record is processed, so are all the other records in your book. It is at the top of the many link branches that comprise a book, the record from which all links point. Record A is an example of a script record.

Figure 38 shows an excerpt from the script record that defines the book *User's Guide to Symbolics Computers*. The table of contents for the book is shown in figure 37. The script record contains the links to the two records that, when processed, will be formatted, labelled, and numbered consecutively as chapters: "Overview of Symbolics Computers" and "Starting up", respectively. This script record is typical, in that it contains only links, no text.

The include links in the "chapter-level" records, in turn, lay out the structure of the chapter. Figure 39 shows the include links for "Overview of Symbolics Computers". When processed, these links become that chapter's sections.

As with the script record, a typical chapter-level record contains little or no text, usually just a brief introduction to orient readers to the sections that follow. Text tends to be concentrated in the lower ends of the link branches, generally in the section-level and subsection-level records. However, this practice is customary

```

|Script "User's Guide to Symbolics Computers"
|Contents
  ➤ Include link to "Overview of Symbolics Computers (section)"
  ➤ Include link to "Starting Up (section)"

|Contents
|Oneliner

|Oneliner
|Keywords

|Keywords
|End of "User's Guide to Symbolics Computers" record

```

Figure 38. The script record sets up the structure of your book.

```

|Chapter "Overview of Symbolics Computers"
|Source Topic
  Overview of Symbolics Computers
|Source Topic
|Contents
  ➤ Include link to "Documentation Notation Conventions (section)"

  ➤ Include link to "Introduction to Genera (section)"

|Contents
|Oneliner
|Oneliner
|Keywords
|Keywords
|End of "Overview of Symbolics Computers" record

```

Figure 39. A sample chapter-level record.

rather than mandatory; Symbolics Concordia imposes no restrictions on mixing text and links.

By repeating this pattern — each level in the hierarchy sets up the next lower level, you can establish the entire structure of the book. Figure 40 illustrates this pattern, from the top level down to the subsection level.

### 13.2.2.3. Advantages of Symbolics Concordia's Linking Strategy

Symbolics Concordia's record-linking strategy produces a very "flexible" book hierarchy. Changing a record's position in the hierarchy entails a straightforward mechanical procedure: Delete the link from the current location and reinsert it in a different record. For example, to make Record C a section rather than a chapter, simply link C from Record B rather than from Record A. To change a record's order in the hierarchy, for example, to have Record E print before Record D, change the order of the links. To drop a record from your book, simply delete all include links to the record. Unlike more traditional documentation systems, it is never nec-

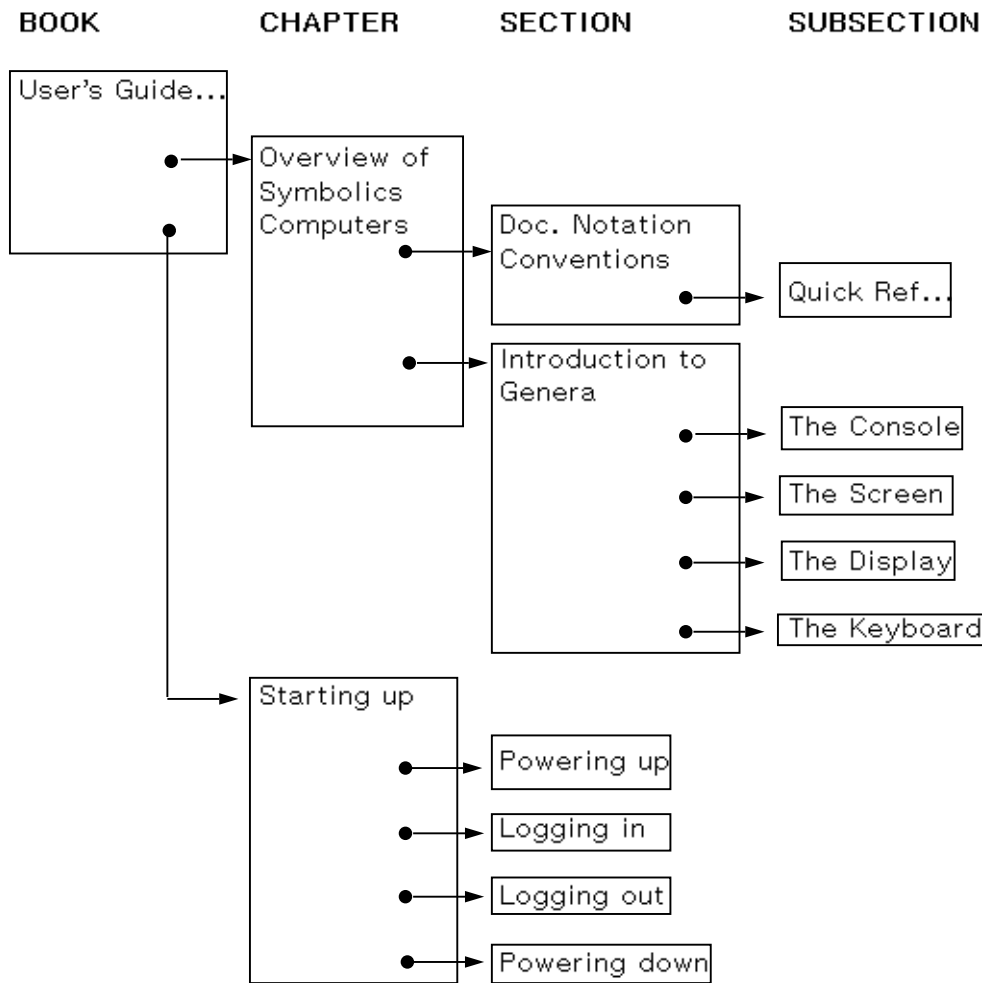


Figure 40. Diagram of book structure.

essary to move or delete chunks of text in a file manually or edit section headings or numbers.

Another advantage of Symbolics Concordia's flexibility is that records can be used in more than one document without relabeling a section as a chapter, or vice-versa, because these things are determined by the linking arrangement in each document.

This method of organizing text compares favorably with conventional systems, in the sense that writers are generally forced to think about organization even as they compose text. In Symbolics Concordia, the writing process is conceptually separated from the job of organizing units of information. You can concentrate on writing (creating records) or on organization (linking records), as you wish.

### 13.2.3. Online Reading Path

The reading path suggested by a printed book is obvious. Its organization is clear. Readers can scan whole chapters, marked as such, to find all the information on one topic. Or they might home in on a small section, flipping back a few pages for context. In short, all the information is available.

Unlike users of conventional paper-only documentation systems, writers using Symbolics Concordia must consider online readers as well. The needs of readers using Document Examiner differ from those reading printed books for two main reasons:

- Document Examiner "flattens" the hierarchy of a printed book; it does not display any visible indication of a record's place in the hierarchy, no numbers or typeface changes make the structure apparent. How conceptually "small" or "large" a topic might be is not obvious, although, one hopes, well-chosen record names will guide the reader. Each record is an equal among records.
- Online readers commonly see topics out of context. When a Document Examiner user issues a Show Documentation command, the formatter processes the selected record and all the links in that record, right on down to the end of the tree. It displays the selected record's name and contents and the correct views of all the links. It is not possible *within the scope of that one command* to see the text in the *parent* record, the record from which the selected record is linked.

**Note:** Several Document Examiner commands, Show Overview and Show Table of Contents, do provide context for the reader, but these are separately invocable from the actual "viewing" commands.

In the absence of visible guideposts, the way in which you link your records becomes especially important, because they significantly affect what your readers see when they look up topics in Document Examiner.

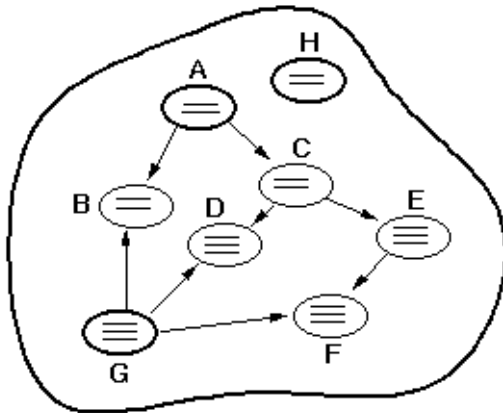
The following table illustrates the difference between using a printed book and using Document Examiner.

<i>Record</i>	<i>Corresponds to a ...</i>
<b>A</b>	Book
<b>B</b>	Chapter
<b>C</b>	Chapter
<b>D</b>	Section
<b>E</b>	Section
<b>F</b>	Subsection

If a reader opens to Chapter C in a printed book, he or she sees the contents of C, D, E, and F. The online viewer of C sees the same sequence of information; the reading path is C, D, E, and F. But when the online viewer asks to see D, he or she will see only D, with no idea of the larger picture provided by C or the additional details given in E. Someone perusing a printed book can easily flip back a few pages for any necessary background information or, conversely, turn the page to keep on reading. However, the online viewer has no guide rail regarding struc-



ture and you, the writer, have no idea at which point in the chain of links, the on-line viewer will choose to begin reading. When inserting links, therefore, you should consider the question: "At any point in the chain of processed links, is the reading path sufficient to give the reader enough *useful* and *usable* information?"



**Document Database**

No hard-and-fast answers can be provided; decisions about which records to link together must be made on an individual basis. However, some alternatives can be offered to help you guide the reader more easily through a topic.

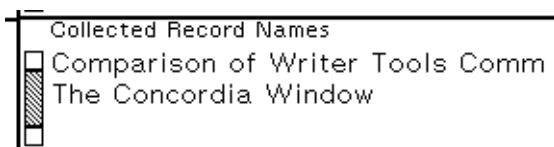
- You should consider changing the link arrangement. You might link E from D rather than from C, if, upon reflection, you decide that the information in E is really indispensable for using the information in D. (In that case, online readers who request D will see D, E, and F.) You might leave the include link as it is, and insert crossreference or topic links. For example, a crossreference or topic reference from D to E would direct the reader's attention to additional information. However, there is a trade-off here: Users who ask to display C will see C, D, E, and F. They might find it peculiar to see a crossreference in D to the very next section, E.

Crossreferences to preceding sections can be even more awkward. For example, you decide that some sort of reference from D to C is necessary, since C provides some useful introductory information. When the reader looks up D in Document Examiner, this arrangement works fine, but when the reader looks up C, such a reference might evoke a rather puzzled reaction: "But I just read this section."

- You might take no action and assume that the reader is familiar with the Show Overview command in Document Examiner, which provides the name of the parent (linking) record and the name of all sibling records (records linked from the same record). Readers can use Show Overview to investigate a topic further.

### 13.3. Using the Collected Record Names Pane

The Collected Record Names pane (shown here) stores record names. Its purpose is to help you create links quickly and easily by collecting, *in advance of making links*, the names of those records that you will link to other records. Names are collected in two ways: (1) Symbolics Concordia automatically adds to the pane the names of all records created in the current boot session, on the assumption that you will soon want to create a link from an existing record to this new record. (2) You can also explicitly add the name of a record to the pane with the "Collect Record Name" command.



When the Create Link command prompts you to specify a record name, simply click Left on a name in the pane. Symbolics Concordia inserts a link in the current record to the selected record. When you can't tell just from the names which record is the right one to choose, then *prior* to creating the link, click  $\mathfrak{m}$ -Left on a name in the pane to display that record's documentation.

After creating the link or when you're sure a record name is no longer needed, you can delete it from the pane by clicking  $\mathfrak{sh}$ -Middle on it. To clear the entire pane, use  $\mathfrak{m}$ -X Clear Record Name Collection.

#### Collect Record Name

Places a record name in the Collected Record Names pane. Collect the names of those records that you want to link to other records. When the "Create Link" command prompts you for a record name, you can click Left on a record name to select it from this pane.

#### Clear Record Name Collection

Removes all record names from the Collected Record Names pane.

Available in the editor as  $\mathfrak{m}$ -X Clear Record Name Collection.

Note that the pane, located in the lower right of the Symbolics Concordia window, is quite small. For information on resizing the pane, see the section "Customizing the Symbolics Concordia Window", page 61.

## 13.4. Link Commands

Symbolics Concordia provides several commands for creating links. The Create Link command makes links *only* to existing records. Writers use Create Link after they have composed and edited several records and then want to organize them into a coherent unit of text.

Other commands offer to create a link as they create a new record. When a region is marked, "Create Record" inserts a new record in the buffer, adds the region to the Contents field, and creates a link in the current record (in which the region is marked) to the new record. Typically, writers use Create Record in this way in order to modularize already existing text. "Create Link and Record" creates a new record and a link to that record from the current record. Typically, writers use Create Link and Record during the initial writing phase, when they realize in advance of writing some material that a new record is needed to maintain modularity and that the text of the new record will flow naturally from the text of the current record.

**Create Link**            Creates a link to an installed record, at the cursor position. The cursor must be within the Contents field of a record. It prompts for a record name. You can click on one of the names in the Collected Record Names pane or type in a name.

It also prompts for a *view*, which determines which record fields get printed or displayed when the record is processed by the formatter. Each type of link produces its own view of a record.

Please read the background information provided about links. See the section "Link Types", page 86.

<i>Type</i>	<i>Description</i>
-------------	--------------------

Include	View = Name and Contents fields.
---------	----------------------------------

Useful for incorporating text into and organizing the structure of a document. It creates visible structure (numbered/labelled sections) in a printed book.

Contents	View = Contents field.
----------	------------------------

Useful for incorporating text into a document. In this respect, it is similar to an include link. The main difference is that it blends text without displaying or printing the name of the record. Unlike an include link, it does *not* contribute to the visible structure of a printed book.

Crossreference	
----------------	--

A crossreference has three *appearances*, Topic, Invisible, and See.

**Topic** Name only. Useful for embedding a topic name in a sentence. You are responsible for spacing around the name.

**Invisible** Does not display. Used in conjunction with the `CrossReference` environment to cause arbitrary pieces of text to be mouse sensitive. See the section "Invisible-style Cross-references", page 91.

**See** Name and type, embedded in the sentence or sentence fragment, "See the *record-type record-name*." You can specify `Initial Cap` for the word "See" and `Final Period` for the period at the end of the sentence by editing the link with `m-x Edit Link` or by clicking `c-m-Right` on the link.

When the record is printed on paper, the crossreference displays the location of the information:

- If the linked record and the linking record are included in the same book, the page number is printed.
- If the linked record and the linking record are *not* included in the same book, the title of the book containing the record is printed.

A crossreference link is useful for directing readers to additional information.

When the record is displayed, the crossreference is mouse-sensitive and looks as follows. Examples:

See the section "Putting Pictures in Text".

See the variable "**cp:prompt\***".

Note that Symbolics Concordia inserts the proper capitalization and punctuation. You, however, have responsibility for correct spacing.

**Precis** View = Name, Arglist (if the object is a Lisp object), and Oneliner fields.

Prints or displays the Lisp object name and argument list on one line and the oneliner summary on the next.

Useful for creating brief, dictionary-like lists or definition tables. Often used with environments to create formatted tables.

Clicking Left or Middle on Create Link in the menu provides a short-cut way to create an include or crossreference link, respectively. Clicking Right displays a menu of all link types.

This command is available as `m-X` Create Link or as Create Link under **Links** in the menu.

You can modify links, using Edit Link.

#### Edit Link

Pops up a menu to change the attributes of the link at the cursor location. You can change the view of the link, the appearance of crossreference links, or even the topic to which the link points.

This command is available as `m-X` Edit Link and by clicking `c-m-Right` on the link to be edited.



## 14. Examining and Changing the Organization of a Document

### 14.1. Examining the Organization of Your Document

From time to time, and certainly before final printing, you will want to verify that the structure of your document is exactly as it should be. For example, you should check that all include links are correct so that your document contains the right text and displays it in the right order. As you revise a document, you need to manage the crossreferences, making sure they are still accurate and useful.

Symbolics Concordia provides commands to help you understand the relationships between records. The Graph Links From Record command shows a visual representation of all the include links from a record. Show Outline displays an even more detailed "table of contents" branching from any record.

Show Links From Record is helpful anytime you want to see how a particular record is related to the other records in your document. It displays a list of all the links from a record, by type.

You should also check that each record that needs a link to it has one. For example, imagine that you have records containing conceptual material and other records documenting related reference information. You might want to check that each "reference" record is linked to the appropriate "conceptual" record. In this case you would use the command Show Links to Record on each reference record.

#### Show Links from Record

Lists links *from* the specified record *to* other records. It prompts for a record name; the default is the current record. The links are listed by type (include, crossreference, and so on).

This command is available as  $\text{M-X}$  Show Links from Record and as Show Links from Record under **Links** in the menu.

#### Show Links to Record

Lists the links *to* a specified record. It prompts for a record name; the default is the current record. The links are listed by type (include, crossreference, and so on).

This command is available as  $\text{M-X}$  Show Links to Record and as Show Links to Record under **Links** in the menu.

#### Graph Links from Record

Shows graphically all the include links *from* the record you specify *to* other records. It prompts for a record name; the default is the last record for which you displayed an overview in the editor or Document Examiner.

In effect, the display shows a table of contents for the specified record: which records are included in a particular record. This command lets you verify the structure of your document, confirming that all records are linked that should be linked.

This command is available as `m-X` Graph Links from Record and as Graph Links from Record under **Links** in the menu.

#### Show Outline

Lists the include links for a record and all the include links in its "included" records. It prompts for a topic name; the default is the current record.

Invoking the command on the *top-level record* (the record that, when displayed, shows all the records in your document) shows you the table of contents for a document.

This command is available as `m-X` Show Outline and as Show Outline under **Topics** in the menu.

## 14.2. Changing the Organization of Your Document

To reorganize a traditional document, you usually have to literally move text around on the screen or, as in the days of pre-computerized paper document preparation, cut and paste pages.

With Symbolics Concordia, you can change the structure of your document by changing a link, changing the order of links in a record, or by removing a link altogether. For example, to delete a section from a chapter, use `c-K` or `c-D` on the link from the chapter to the section. To change the order of sections within a chapter, again use `c-K` or `c-D` to remove a link and `c-Y` to yank it back into the record where you want it. (See **Kill History** in the menu.)

Restructuring your document might require you to change either the name or type of a link. For example, you decide to crossreference a different record than the one referred to currently, or you decide that a crossreference to a record will suffice where you currently include the text of that record. In each case it is not necessary to delete the link and create a new one; simply edit the link name and/or type.

The procedure is simple. Click `c-m-Right` on a link. A menu pops up, as in Figure 41, which gives the link type and target. The current link type, contents, is bold-faced. To change the link type, click on a new link type. To change the target of the link, click on the name and type in the name of an existing record. When you're satisfied, press `END`. The link is updated in the record.



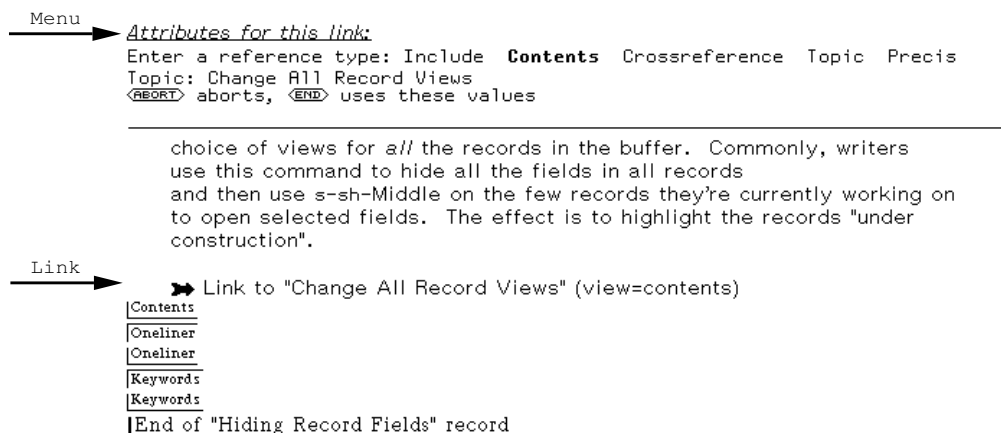


Figure 41. A menu appears when you edit a link type or target.



## 15. Controlling Your Document's Appearance

You can control the appearance of your book in two major ways:

- By inserting markup
- By changing the typeface in which text is rendered

### 15.1. Introduction to Symbolics Concordia Markup Language

The purpose of Symbolics Concordia markup language is to help the writer produce a finished document that looks good. The language described here should be distinguished from the *overall* book design facility that allows the documentation to be published in paper form. This facility specifies high-level page and book design (for example, margins or the appearance of the table of contents) and is usually under the control of a professional book designer or book production administrator. (See the section "Guide to Symbolics Concordia Book Design", page 357.)

Your contribution to the appearance of your book is still substantial, however. Symbolics Concordia's markup language provides simple means for formatting text and illustrations in a variety of ways and controlling some aspects of page layout. You insert the markup directly into the record. When the record is processed by the formatter, which prepares the record for online display or printing, the instructions specified by the markup are executed. For example, when the formatter encounters the markup for a bulleted list, it formats the marked-up text as a bulleted list. The markup does not appear in the display. Or when the formatter sees markup in a record that means "leave three lines of white space here", the finished document contains the requested amount of space.

Symbolics Concordia's markup language consists of *environments* and *commands*.

#### 15.1.1. Environments and How to Use Them

*Environments* provide a mechanism for creating tables, lists, examples, headings, and other kinds of text formats. The Symbolics Concordia formatter processes the text within an environment in the way specified by the definition of the environment (set globally by your site's book design standards). For example, text within an *Itemize* environment produces a bulleted list, when the record is displayed online or printed on paper. The same text placed in an *Enumerate* environment produces a numbered list.

##### 15.1.1.1. Characteristics of Mark-up Environments

Environments can be *filled* or *unfilled*. A filled environment is one in which text extends to the right-hand margin. Text in the environment is added to the line until it is "filled." Some examples of filled environments are *Enumerate* and *Description*. In unfilled environments, lines of text appear as they are typed in that environment. Some examples are: *Format*, *Display*, and *Center*.

The markup for an environment is shown in the buffer as small iconic boxes. The following diagram shows the markup for the Itemize environment.

```

┌Itemize
└Itemize

```

Environments can also be nested one inside another. For example, an Itemize nested within an Enumerate produces a list whose major items are numbered and some of whose subitems are bulleted.

In addition to text, environments can contain Symbolics Concordia markup commands, links, and the representations of Graphic Editor illustrations.

### Run-in Markers

Symbolics Concordia markup diagrams appear on lines by themselves. This is necessary because text strings and diagrams cannot be mixed on the same line. This introduces a problem with newlines.

A superscript like this:

$$A^2$$

looks like this on your screen when you are editing the record:

```

  A
  ⋄+
    2
  ⊥+

```

*Run-in markers* (^ and v) indicate that there is no newline or RETURN character following A in the buffer. The display of the A and its superscript will be right next to each other.

If this superscripted A looked like this:

```

  A
  |+
    2
  ⊥+

```

It would display like this:

$$A^2$$

Notice that there is a space between the A and the superscript. There are no run-in markers in the source, indicating that there is a real RETURN or newline character in the buffer after the A. The formatter always turns a single RETURN into a

space for formatting purposes, so there is a space between the A and the superscripted 2.

You can remove the RETURN by deleting it from the end of the line with the A on it. This puts a run-in marker back on the markup.

Here is another example:

This enumerate environment has run-in markers on the multiple:

```

Enumerate
  Item one.

  Item two.
  ◊Multiple
    Item three.

    The second paragraph of Item three.

  ◊Multiple
Enumerate

```

Not surprisingly, it displays with Item two and Item three run together, without any space:

1. Item one.
2. Item two.Item three.

The second paragraph of Item three.

We can remove the run-in markers by adding a RETURN character after "Item two."

```

Enumerate
  Item one.

  Item two.
  Multiple
    Item three.

    The second paragraph of Item three.

  Multiple
Enumerate

```

The RETURN character adds a space between "Item two" and "Item three," but these lines are still run together when displayed since there is only a single RETURN character between Item two and the Multiple.

1. Item one.

2. Item two. Item three.

The second paragraph of Item three.

Items in an Enumerate environment must be separated by a paragraph break, which is two RETURN characters in succession. If we add a RETURN after Item two, like this:

```

Enumerate
  Item one.

  Item two.

Multiple
  Item three.

  The second paragraph of Item three.

Multiple
Enumerate

```

It now displays the way we want it:

1. Item one.
2. Item two.
3. Item three.

The second paragraph of Item three.

### 15.1.1.2. Creating an Environment

To insert markup for an environment into a record, press `Ⓢ-M`, use `m-X` Create Environment, or click on Create under **Markup** in the menu. If you need assistance, press `HELP` to see a complete list of available environments. The command inserts the markup for the environment in your buffer.

Insert the text you want to format within the markup. Alternatively, you can type the text first, mark the text as a region, and then insert the markup in the same way. The markup will be inserted around the region.

Note that each environment has its own "rules and regulations" for use. For example, Itemize requires a blank line between individual entries in the list. Any special requirements are documented in the descriptions of individual environments. See the section "Concordia Markup Language", page 119..

#### Create Environment

Inserts the markup for a formatting environment into a record, placing each delimiter on a new line. When a region is marked, it inserts the markup around the region. It prompts

for an environment name. When you invoke the Create Environment command, the default is the last environment selected. Press **HELP** to see a list of the available environments.

This command is available as `m-X` Create Environment, Create under **Markup** in the menu, or as `s-M`.

### 15.1.1.3. Environment Attributes

Each environment has *attributes*, or characteristics, that determine exactly how text in that environment looks. Attributes control such things as how much blank space precedes and follows the environment, how much blank space is left between each element in the environment, and so on.

The attributes of each environment have certain default values. For example, by default Itemize leaves one blank line *above* the first item, one blank line *below* the last item, and one blank line *between* items. Thus, the default value of Above is 1 line; Below, 1 line; Spread, 1 line.

For more information on how to change the attributes of an instance of an environment, see the section "Modifying a Single Instance of an Environment", page 389.

### 15.1.1.4. Killing an Environment and Environment Markup

To remove both the markup and contents of an environment, use Kill Environment. To remove just the markup, use Remove Markup.

**Kill Environment** Deletes the nearest environment markup enclosing the cursor, along with its contents, and places it on the kill ring. If the environment contains text and/or markup, it highlights the environment and asks for confirmation.

You can yank back what you have killed. See the commands under **Kill History** in the menu.

**Note:** If you need to delete only the markup and not the contents of the environment, use "Remove Markup".

The command is available as `s-K` or as Kill under **Markup** in the menu.

**Remove Markup** Deletes the nearest environment markup enclosing the cursor and places it on the kill ring. If the markup contains text and/or markup, it highlights the environment and asks for confirmation.

**Note:** This command does not delete the contents of the environment, either text or other markup; to remove both the markup and its contents, use "Kill Environment".

You can yank back markup. See the commands under **Kill History** in the menu.

The command is available as  $\text{⌘-}^{\wedge}$  or as Remove Markup under **Markup** in the menu.

### 15.1.2. Markup Commands and How to Use Them

In addition to environments, Symbolics Concordia markup language also has a set of *commands*, which are instructions to the formatter. These commands control some basic elements of page layout, like line and paragraph breaks and tab stops. Commands can, and sometimes *must*, be used within environment markup.

#### 15.1.2.1. Inserting a Markup Command

To add a formatting command, press  $\text{⌘-M}$ , use  $\text{⌘-X}$  Create Command, or click on Create under **Markup** in the menu. If you need assistance, press HELP to see a complete list of available commands.

Symbolics Concordia inserts the appropriate markup for the command in the buffer; for almost all commands, the markup is a pointing index finger followed by the command name, such as:

 ignore-white-space

The em dash is an exception; the dash character itself is inserted directly into the buffer.

Some commands require you to enter a value — numbers or text. For example, when you select the Blankspace command, a menu prompts you to specify the amount of vertical space to leave blank, such as 2 lines or 3 inches.

**Create Command** Inserts markup for a formatting command into a record. The markup, usually a pointing index finger followed by the command name, is placed on a new line.

It prompts for a command name; the default is the last command selected. Press HELP to see a list of the available commands.

If the command takes an argument, a menu pops up and prompts you to supply a value for the argument, for example, a vertical distance for the Blankspace command, or a phrase or sentence for the Caption command.

This command is available as Create or  $\text{⌘-M}$  under **Markup** on the menu or as  $\text{⌘-X}$  Create Command.

### Command Keyboard Accelerators

Symbolics Concordia supports the following keyboard accelerator commands:

$\text{⌘-TAB}$	Inserts an icon that signifies a tab character.
$\text{⌘->}$	Inserts an icon that signifies that text should be positioned at the right margin.



<code>s-=</code>	Inserts an icon that signifies that text should centered within a column.
<code>s-</code>	Inserts an em dash.
<code>s-L</code>	Makes a Lisp language form.

### 15.1.2.2. Changing a Markup Command

To change the value(s) associated with a command, click `s-sh-M` on the command markup. This changes the view of the markup, presenting its values in a form you can edit in the Symbolics Concordia editor buffer.

### 15.1.2.3. Killing a Markup Command

Use `c-K` or `c-D` to kill a formatting command. You can yank back killed commands. See **Kill History** in the menu.

### 15.1.3. Internal and External Markup

Additional markup commands and environments are available for use primarily in book design. (See the section "Book Design Functions Dictionary", page 418.) Many of these *internal* commands and environments have hyphenated names and, since they are intended for use in book design, they cannot be inserted in a .sab file using Create Markup. However, sometimes it is necessary to use one of them. If you need to insert an internal command or environment in a .sab file, you can insert it using `c-U s-M`.

## 15.2. Concordia Markup Language

This section describes Concordia's markup language, which consists of environments, environment attributes, and commands. These terms are defined in "Introduction to Symbolics Concordia Markup Language".

The first section illustrates the use of some commonly used environments and is organized by function: how to make lists, how to set off text from surrounding text, and how to position text within the printing column.

The next sections describe some common environment attributes and markup commands. The final sections list and give a brief description of Concordia markup environments and commands and Concordia attributes.

### 15.2.1. Some Commonly Used Environments

#### 15.2.1.1. Making Lists in Concordia

Itemize, Enumerate, and Checklist are environments for creating bulleted, numbered, and checked lists.

Description can format text in list-like ways, make side-by-side lists, or list individual words, phrases, or technical terms with adjacent paragraphs of explanation.

## Itemize Environment

Itemize is a "filled environment" that produces a bulleted list. The items in the list are marked by bullets, or tick-marks. Separate each item by one blank line.

The Enumerate and Checklist environments are similar to Itemize.

The following example shows the use of the Itemize environment with its default values in effect.

### Markup

```

Example
Remember to bring:
◊Itemize
    suntan lotion

    a beach towel

    a trashy novel
◊Itemize
The flight leaves at 10:00. Be at the airport two hours early.
^Example

```

### Display

```

Remember to bring:

• suntan lotion

• a beach towel

• a trashy novel

The flight leaves at 10:00. Be at the airport two hours early.

```

You produce a tighter- or looser-looking list by changing the distance between items and the distance between the list and surrounding text. Use the "Spread Attribute", the "Above Attribute", and the "Below". Here, Spread, Above, and Below are set to zero.

### Markup

```

Attributes for this environment:
Environment Name: Itemize
Above: 1 Lines
Below: 1 Lines
BlankLines: Break Hinge Hingebreak Hingekeep Ignore Ignored Kept
Indent: -2 Characters
LeftMargin: +2 Characters
Spacing: 1 Lines
Spread: 0 Lines
Other Attribute: an attribute name
Click on this line to reset attributes to default values.
<ABORT> aborts, <END> uses these values

```

*Display*

Remember to bring:

- suntan lotion
- a beach towel
- a trashy novel

The flight leaves at 10:00. Be at the airport two hours early.

You can change the position of an itemized list within the column by changing the "Leftmargin Attribute". Don't try to position the Itemize environment by nesting it in a Center Environment — the tick-marks won't be aligned. In the next example, the Leftmargin attribute is set to +6 characters.

*Markup*

```

Attributes for this environment:
Environment Name: Itemize
Above: 1 Lines
Below: 1 Lines
BlankLines: Break Hinge Hingebreak Hingekeep Ignore Ignored Kept
Indent: -2 Characters
LeftMargin: +6 Characters
Spacing: 1 Lines
Spread: 1 Lines
Other Attribute: an attribute name
Click on this line to reset attributes to default values.
<ABORT> aborts, <END> uses these values

```

*Display*

Remember to bring:

- suntan lotion
- a beach towel
- a trashy novel

The flight leaves at 10:00. Be at the airport two hours early.

The "Indent" controls the distance between the bullet, or tick-mark, and the text. This is an example of a value of -5 characters for Indent.

### Markup

```

]Attributes for this environment:
Environment Name: Itemize
Above: 1 Lines
Below: 1 Lines
BlankLines: Break Hinge Hingebreak Hingekeep Ignore Ignored Kept
Indent: [-5 Characters]
LeftMargin: +6 Characters
Spacing: 1 Lines
Spread: 1 Lines
Other Attribute: an attribute name
Click on this line to reset attributes to default values.
<ABORT> aborts, <END> uses these values

```

### Display

Remember to bring:

- suntan lotion
- a beach towel
- a trashy novel

The flight leaves at 10:00. Be at the airport two hours early.

The Itemize environment can be nested within itself many times, creating alternating solid and hollow tick-marks.

- item one, level one
  - item one, level two
    - item one, level three
      - item one, level four
    - item two, level three
  - item two, level two
- item two, level one

### Enumerate Environment

Enumerate is a "filled environment" that produces a numbered list. It is very similar to the "Itemize Environment" and the "Checklist Environment".

Using the default attributes, the Enumerate environment generates a simple numbered list.

Nested Enumerate environments produce lists that are lettered and numbered, and so are commonly used to make outlines.

These examples show how to use the Enumerate environment.

#### *Markup*

```
|Enumerate
  George Washington
    John Adams
      Thomas Jefferson
|Enumerate
```

#### *Display*

1. George Washington
2. John Adams
3. Thomas Jefferson

This example shows how to use nested Enumerate environments.

*Markup*

```
Enumerate
  Overview of Symbolics Computers
  Enumerate
    Documentation Notation Conventions
    Enumerate
      Notation Conventions Quick Reference
    Enumerate

  Introduction to the Symbolics 3600 Family of Computers

  Enumerate
    Introduction

    Packaging
  Enumerate

  Introduction to Genera

  Enumerate
    The Console

    The Screen
  Enumerate

  Enumerate

  Starting up
Enumerate
```

*Display*

1. Overview of Symbolics Computers
  - a. Documentation Notation Conventions
    - i. Notation Conventions Quick Reference
  - b. Introduction to the Symbolics 3600 Family of Computers
    - i. Introduction
    - ii. Packaging
  - c. Introduction to Genera
    - i. The Console
    - ii. The Screen
2. Starting up

The Enumerate environment can be altered by changing its default attributes. Refer to "Itemize Environment" to see typical alterations.

### Checklist Environment

A filled environment which produces a list marked by pointer icons on the screen. There are no markers when this environment is hardcopied.

Checklist is similar to the Itemize environment and Enumerate environment.

Using the default attributes, the Checklist environment generates a simple list.

*Markup*

```
|Checklist
  eggs
    bacon
      toast
|Checklist
```

*Display*

<input type="checkbox"/>	eggs
<input type="checkbox"/>	bacon
<input type="checkbox"/>	toast

The Checklist environment can be altered by changing its default attributes. Refer to "Itemize Environment" to see typical alterations.

### Description Environment

Description is a "filled environment" that has a wider left margin after the first line. Because it resembles a two-column table, Description is often used to format lists of definitions. It is also useful for displaying paragraphs of text with headers.

You must insert a tab character to separate the header from the body of the paragraph. Press `s-TAB` or use the Tab-to-tab-stop command.

#### Markup

```

Description
  Thing One↵ This is a description of Thing One. You can
  nest Description environments for lists within lists.

  ↵ Begin multiple paragraphs for an item with s-Tab or
  enclose all of the paragraphs for the item within a
  Multiple environment.
Description

```

#### Display

<p>Thing One</p>	<p>This is a description of Thing One. You can nest Description environments for lists within lists.</p> <p>Begin multiple paragraphs for an item with <code>s-Tab</code> or enclose all of the paragraphs for the item within a Multiple environment.</p>
------------------	--



### 15.2.1.2. Setting Off Text in Concordia

You can set off text by using the Format, Verbatim, Display, or Example environments, by changing the character style of a passage, or by using special symbols in your text.

#### Format Environment

Format is an "unfilled environment" that produces text exactly as you type it using the current font, without moving margins or justifying text. It is used to show unusual formatting.

```
See          anywhere.  
  how      text  
    you    position  
      can
```

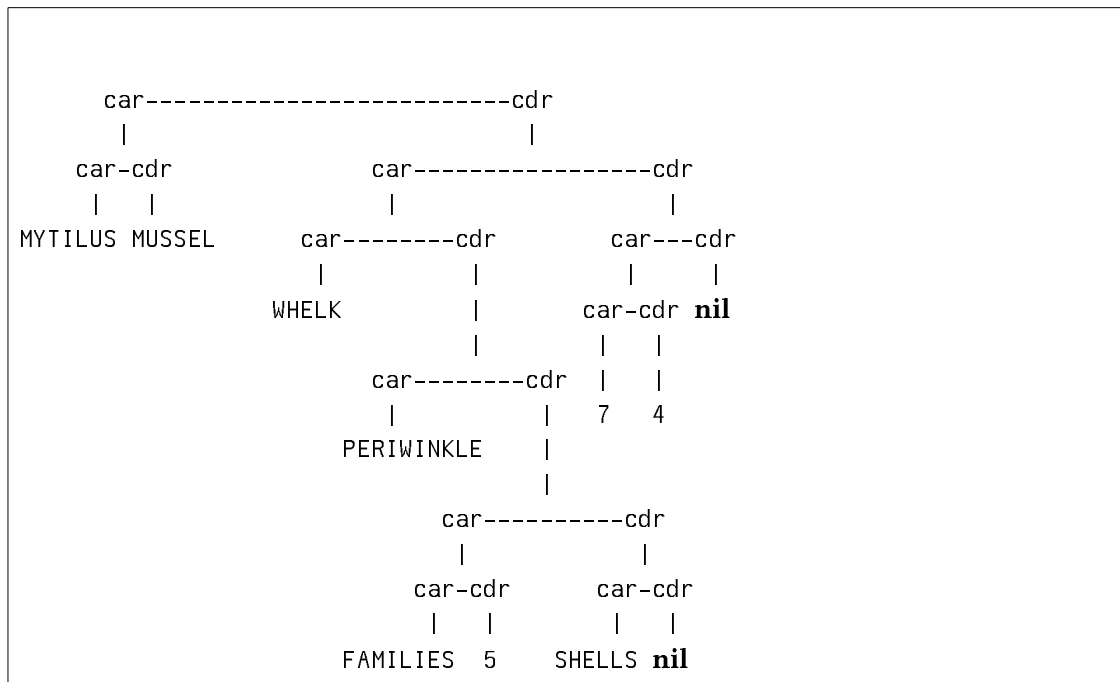
#### Verbatim Environment

Verbatim produces everything you type exactly as you type it, without moving margins or justifying text. It is used to show unusual formatting.

Verbatim is an "unfilled environment" like "Format Environment" except that Verbatim uses a fixed-width font unlike regular text.

```
See          anywhere.  
  how      text  
    you    position  
      can
```

You can also use Verbatim to make tables and figures.



### Display Environment

Display is an "unfilled environment" where each line in the Concordia file produces one line in the processed text. Both the left and the right margins are widened.

It is often used to show lists without tick-marks or numbers.

```

This text is in a Display environment.
This is a new line.

```

### Example Environment

Example is an "unfilled environment" that creates a region with a wider margin and that uses a fixed-width font. It is designed to show examples of computer interaction.

Often, examples of actual programs contain lines that are too long to be printed on one line in the Example environment. Refer to the section "Longlines Attribute" for more information.

```
(setq b 1)
'(a b c) => (a b c)
'(a ,b c) => (a 1 c)
'(abc ,(+ b 4) ,(- b 1) (def ,b)) =>
      (abc 5 0 (def 1))
```

### 15.2.1.3. Positioning Text in Concordia

You can change the position of your text by inserting it in an environment, creating tabs within your text, or changing the attributes of an environment.

Center, Flushleft, and Flushright are three common environments for changing the position of your text in a displayed record.

#### Center Environment

An "unfilled environment" that causes each line within it to be centered in the displayed record.

#### Flushleft Environment

Flushleft environment is an "unfilled environment" that causes each line within it to be aligned with the displayed record's left "global margin". The distance to the margin can be changed by altering the "Leftmargin Attribute".

```
This text is in a Flushleft environment.
This is a new line.
```

#### Flushright Environment

Flushright environment is an "unfilled environment" that causes each line within it to be aligned with the displayed record's right "global margin". The distance to the margin can be changed by altering the "Rightmargin Attribute".

```
This text is in a Flushright environment.
This is a new line.
```

## Setting and Using Tabs in Concordia

You can use the following commands to create tabs:

TabClear Command	Eliminates any tabs that were previously set in a file.
TabDivide Command	Divides the formatting region into columns.
TabSet Command	Creates tabs within the text at user-selected distances.
<code>␣-TAB</code> Command	Moves text to the right until it encounters the next tab setting (that is, performs a tab-to-tab stop).
<code>s=</code> Command	Centers text between tabstops. It is frequently used to create columns of centered text.
<code>s-&gt;</code> Command	Moves text, on the same line as other text, against the right margin.

### TabClear Command

Eliminates any tabs that were previously set in a file. Usually used before any type of tab setting command, like "TabDivide Command" or "TabSet Command".

### TabDivide Command

Divides the formatting region into columns. Columns are of equal widths and sizes to fill the region horizontally. Text items are separated for formatting by using the "`␣-TAB` Command" or by pressing `␣-TAB`.

This command takes an integer as an argument.

Although it is not necessary to clear tabstops within an environment, doing so guarantees that the TabDivide command will work as you expect it to. It is a good work habit to put a TabClear command before each TabDivide command you use.

When equally sized columns are not desired, use the "TabSet Command".

Wrap tabbed text in a Format Environment to override any formatting defaults.

This is an example of how to set up a two-column table.

*Markup*

*Wrappers* ↔ *Whoppers*

Similar to a macro. ↔ Similar to a function  
 If a wrapper is modified, all ↔ If a whopper is modified,  
 Combined methods using it must ↔ only the whopper must be  
 be recompiled (this is done ↔ recompiled.  
 automatically.)

*Display**Wrappers*

Similar to a macro.  
 If a wrapper is modified, all  
 Combined methods using it must  
 be recompiled (this is done  
 automatically.)

*Whoppers*

Similar to a function  
 If a whopper is modified,  
 only the whopper must be  
 recompiled.

This is an example of how to set up a three column table.

*Markup*

☞ TabDivide 3

*Character Style* ↔ *Default* ↔ *Result of*  
*of a Character* ↔ *Character Style* ↔ *Merging*

NIL.NIL.NIL ↔ FIX.ROMAN.NORMAL ↔ FIX.ROMAN.NORMAL  
 NIL.ITALIC.LARGE ↔ FIX.ROMAN.NORMAL ↔ FIX.ITALIC.LARGE  
 NIL.ITALIC.SMALLER ↔ FIX.ROMAN.NORMAL ↔ FIX.ITALIC.SMALL  
 SWISS.BOLD.LARGER ↔ FIX.ROMAN.NORMAL ↔ SWISS.BOLD.LARGE

*Display*

<i>Character Style of a Character</i>	<i>Default Character Style</i>	<i>Result of Merging</i>
NIL.NIL.NIL	FIX.ROMAN.NORMAL	FIX.ROMAN.NORMAL
NIL.ITALIC.LARGE	FIX.ROMAN.NORMAL	FIX.ITALIC.LARGE
NIL.ITALIC.SMALLER	FIX.ROMAN.NORMAL	FIX.ITALIC.SMALL
SWISS.BOLD.LARGER	FIX.ROMAN.NORMAL	SWISS.BOLD.LARGE

## TabSet Command

Creates tabs within the text at distances you specify.

- Use a horizontal distance: number of characters, inches, or centimeters.
- Use the "s-TAB Command" or s-TAB to separate text items.
- Insert "TabClear Command" to clear previously set tabs.

It is a good work habit to put a TabClear command at the end of any environment where tabs are explicitly set. Clearing tabs that are no longer necessary, guarantees that the TabSet command will work as you expect it to.

Wrap tabbed text in a Format Environment to override any formatting defaults in the active environment.

*Markup*

```

Format
  ⌘ TabClear
  ⌘ Tabset, Tabs 1.5 Inches, 3.25 Inches, and 4 Inches
  array[→ vector[→ list[→ symbol
  simple-array[→ bit-vector[→ cons[→ null
  simple-vector[→ simple-bit-vector[→ [→ keyword
  structure
Format
...

```

Notice that a column may be left blank by inserting two tab icons in a row.

*Display*

array	vector	list	symbol
simple-array	bit-vector	cons	null
simple-vector	simple-bit-vector		keyword
structure			

### s-TAB Command

Moves text to the right until it encounters the next tab setting (that is, performs a tab-to-tab stop). It appears on the screen as this icon:

↳

Use `s-TAB` in the "Description Environment" to separate the header from the paragraph.

#### Markup

<u>Description</u>	
	<code>stack-type</code> ↳ The type of the stack. Enter Control, Binding, or Data.
	<code>stack-size</code> ↳ The size of the stack.
	↳ Enter a number of machine words that represents the stack size.
<u>Description</u>	

#### Display

<code>stack-type</code>	The type of the stack. Enter Control, Binding, or Data.
<code>stack-size</code>	The size of the stack.
	Enter a number of machine words that represents the stack size. The following example shows the use of the Itemize environment with its default values in effect.

**s-= Command**

Centers text between tabstops. It is frequently used to create columns of centered text.

It appears on the screen as this icon:

⇆

Refer to "TabSet Command" and "TabDivide Command" for more information about setting tabs.

*Markup*

```

⇆ TabDivide 4
⇆ Year ⇆ Party ⇆ President ⇆ Vice-President ⇆

⇆ 1960 ⇆ Democrat ⇆ Kennedy ⇆ Johnson ⇆
⇆ 1964 ⇆ Democrat ⇆ Johnson ⇆ Humphrey ⇆
⇆ 1968 ⇆ Republican ⇆ Nixon ⇆ Agnew ⇆

```

*Display*

<i>Year</i>	<i>Party</i>	<i>President</i>	<i>Vice-President</i>
1960	Democrat	Kennedy	Johnson
1964	Democrat	Johnson	Humphrey
1968	Republican	Nixon	Agnew

**s-> Command**

Moves text, on the same line as other text, against the right margin. It appears on the screen as this icon:

→]

When there is too much text on one line to fit everything after the icon, the text following the icon is placed on the following line.



*Markup*

SYMBOLICS, Inc. ↵Cambridge, MA
--------------------------------

*Display*

SYMBOLICS, Inc.	Cambridge, MA
-----------------	---------------

**Creating Simple Tables**

The SimpleTable environment allows you to create simple tables. It is an unfilled environment, so each line in the table corresponds to a line in your buffer. Columns are separated by tabs. The number of columns (as determined by the tabs in each line) and how much text is in each column, are used to calculate column width based on the longest lines in the table. Tabstops are set for you by the SimpleTable environment. You can change character styles (bold, italic) but you cannot use `␣-L` markers or links inside a SimpleTable environment.

**Note:** Simple Tables do not display correctly in the Page Previewer. They take up the appropriate amount of space on the page, but because the Page Previewer does not scale fonts, the text appears to overlay the rules. They come out correctly in hardcopy.

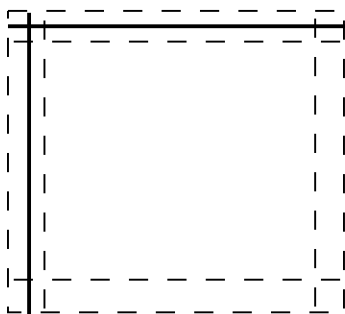
The SimpleTableSpecs command specifies horizontal and vertical rules for a simple table.

The possible horizontal rules start above the first line of the table (position 0). The possible vertical rules start to the left of the first column (position 0). You must specify each rule you want explicitly (remember, this is a SimpleTable), so if you have a three column table that is three lines long, and you want rules around all sides and between each line and column, you would specify:

```
␣SimpleTableSpecs Hrules 0,1,2,3; Vrules 0,1,2,3
```

You enter the SimpleTableSpecs with `␣-M`.

You probably should not make rules around an entire SimpleTable because of the way that the individual "cells" of the table work. Each table entry lives in a cell. The cells have a border around them and the rules are drawn centered in the border. This is fine for the interior walls of cells, but if you want your corners on the outside to meet it won't work. Here's an upper left corner cell with rules at 0,0:



*Markup*

```

Simpletable
  ↪ SimpleTableSpecs Hrules 1; Vrules 1,2
  thing described↪ its discussibility↪ what it means
  :this↪ at hand↪ what this means
  :that↪ remote↪ What that means
Simpletable

```

*Display*

<i>thing described</i>	<i>its discussibility</i>	<i>what it means</i>
<b>:this</b>	at hand	what this means
<b>:that</b>	remote	What that means

Empty cells are ignored, so if you want a gap in your table (say, if you wanted "at hand" above to be a blank), you would place a space between the two tabs in the input.

You can omit the SimpleTableSpecs and have no rules. However, if you supply SimpleTableSpecs you must supply at least one vertical and one horizontal rule. You do not have to specify a continuous series of rules. For example, if you have an item in your table that runs onto two lines, you can skip that position in the horizontal rules so that both lines of the item end up in the same "box". This SimpleTable has horizontal rules at positions 1, 2, and 5:

This	Here
That	There
The Other Thing That Takes More Than One Line	Way Over Yonder
Another One	Nearby

### 15.2.2. The Most Common Environment Attributes

You can also alter the way your text looks within an environment by changing its attributes. In order to see the attribute menu for an environment, place your mouse cursor on either the top or bottom delimiter of an environment and press `C-M-Right`.

These are the most commonly used Symbolics Concordia environment attributes:

Above Attribute	Determines the amount of white space preceding the environment.
Below Attribute	Determines the amount of white space following the environment.
Blanklines Attribute	Determines the treatment of blank lines within the environment.
Group Attribute	Causes the text within the environment to be grouped together and to appear all on one page. If the grouping is too large to fit on the current page, a new page will be started, leaving white space at the bottom of the previous page. An environment can use one of the attributes: <code>Float</code> , <code>FloatPage</code> , <code>Free</code> , or <code>Group</code> .
Indent Attribute	Controls the displacement of the first line of each paragraph in the environment from the left margin specified in the document type.
Leftmargin Attribute	Sets the width of the left margin.
Longlines Attribute	Specifies what happens to lines that would extend past the right margin of the current environment. Valid only for unfilled environments.
Rightmargin Attribute	Sets the width of the right margin.
Spread Attribute	Specifies the amount of space added to the <code>Spacing</code> attribute value to determine the vertical space between paragraphs.

**Spacing Attribute** Specifies baseline-to-baseline vertical spacing.

For a complete description of these commands, see the section "Dictionary of Symbolics Concordia Markup Environments and Commands", page 141.

### 15.2.3. The Most Common Markup Commands

Use the Caption, Tag, and Ref commands to refer to Symbolics Concordia illustrations, tables and figures.

*Markup*

```
Figure
⌘ Ref, Code horizontal-bug
shows a bug.
|Figure ...
  ⌘ Picture "bug-4" from file SCRC|SAP:>sys>{
    ⌘ Caption A Bug
    ⌘ Tag, Code horizontal-bug
|Figure ...
```

*Display*

Figure 1 shows a bug.

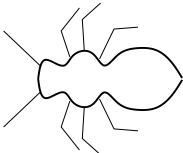


Figure 1. A Bug

**Caption Command** Specifies the caption for a figure or table.

**Tag Command** Specifies a *Codeword* as a crossreference label showing the position and number of an equation, theorem, figure, or table. Used with the Caption command.

**Ref Command**      Retrieves and prints the value associated with a counter used to count page numbers, figures, tables, chapters, sections, and appendixes.

For a complete description of these commands, see the section "Dictionary of Symbolics Concordia Markup Environments and Commands", page 141.

#### 15.2.4. Specifying Page Headings and Footings

You can specify running heads for your document by placing a Pageheading command in your top level record. Create Markup Pageheading requests a pageheading command. A template pops up to allow you to insert the information to be printed.

```

|PageHeading
  Even: Yes  No
  Odd:  Yes  No
  Immediate: Yes  No
  |Left
  |Left
  |Center
  |Center
  |Right
  |Right
  |Line
  |Line
|PageHeading

```

The fields in the template are:

- |                         |   |
|-------------------------|---|
| Even and Odd            | Whether this heading information is for an even page (lefthand or <i>verso</i> ) heading or an odd page (righthand or <i>recto</i> ) heading. You need two pageheading commands, one with <b>yes</b> for Even and the other with <b>yes</b> for Odd.  |
| Immediate               | Whether the headings are to take effect immediately, on the current page, or not until the second page. If you want the headings on the first page to differ from those on the second and subsequent pages, you need two pageheading commands, one with Immediate <b>yes</b> , followed by another with Immediate <b>no</b> . |
| Left, Center, and Right | The three text areas in the heading. You insert the   |

text you want to appear in each place. To get the page number, insert the markup `Value` and specify the counter `Page`.

**Line** An additional line of text, appearing under the left heading and possibly extending across the top of the page. If you place the command `Replicate-Pattern` followed by an underscore, you get a bar across the page. (Note: you insert `Replicate-Pattern` with `c-U s-M`, since it is an internal command.)

To insert literal spaces into a heading, for instance to specify that you want the current chapter title followed by 4 spaces and then the page, use the markup `Hsp` (horizontal space) which takes a horizontal distance as an argument.

For example:

```

⌘value, Name chapter
⌘hsp, Distance 4 Characters
⌘value, Name page

```

Here is what the command templates might look like to print headings for this section:

<u>PageHeading</u>	<u>PageHeading</u>
Even: <b>Yes</b> No	Even: Yes <b>No</b>
Odd: Yes <b>No</b>	Odd: <b>Yes</b> No
Immediate: <b>Yes</b> No	Immediate: <b>Yes</b> No
<u>Left</u>	<u>Left</u>
⌘Value, Name page	⌘Value, Name sage::section
<u>Left</u>	<u>Left</u>
<u>Center</u>	<u>Center</u>
<u>Center</u>	<u>Center</u>
<u>Right</u>	<u>Right</u>
⌘Value, Name sage::chapter	⌘Value, Name page
<u>Right</u>	<u>Right</u>
<u>Line</u>	<u>Line</u>
<u>Line</u>	<u>Line</u>
<u>PageHeading</u>	<u>PageHeading</u>

Page footings are specified similarly.

Page headings and footings have no effect online.

### 15.2.5. Dictionary of Symbolics Concordia Markup Environments and Commands

`s- _` Creates an em dash, the kind of hyphen used to give emphasis or to expand on an idea in the main clause of a sentence — like this.

Unlike other commands, the actual character is inserted in the buffer, not the markup for it.

`s- =` Centers text between tabstops. It is frequently used to create columns of centered text.

It appears on the screen as this icon:

⇄

Refer to "TabSet Command" and "TabDivide Command" for more information about setting tabs.

*Markup*

```
⇄ TabDivide 4
⇄ Year ⇄ Party ⇄ President ⇄ Vice-President ⇄
⇄ 1960 ⇄ Democrat ⇄ Kennedy ⇄ Johnson ⇄
⇄ 1964 ⇄ Democrat ⇄ Johnson ⇄ Humphrey ⇄
⇄ 1968 ⇄ Republican ⇄ Nixon ⇄ Agnew ⇄
```

*Display*

<i>Year</i>	<i>Party</i>	<i>President</i>	<i>Vice-President</i>
1960	Democrat	Kennedy	Johnson
1964	Democrat	Johnson	Humphrey
1968	Republican	Nixon	Agnew

`s- >` Moves text, on the same line as other text, against the right margin. It appears on the screen as this icon:

→]

When there is too much text on one line to fit everything after the icon, the text following the icon is placed on the following line.

### *Markup*

```
SYMBOLICS, Inc. ↪Cambridge, MA
```

### *Display*

```
SYMBOLICS, Inc. Cambridge, MA
```

`␣-TAB`

Moves text to the right until it encounters the next tab setting (that is, performs a tab-to-tab stop). It appears on the screen as this icon:

↪

Use `␣-TAB` in the "Description Environment" to separate the header from the paragraph.

### *Markup*

```

Description
  stack-type↪ The type of the stack. Enter Control, Binding, or Data.
  stack-size↪ The size of the stack.
  ↪Enter a number of machine words that represents the stack size.
Description

```

### *Display*



<i>stack-type</i>	The type of the stack. Enter Control, Binding, or Data.
<i>stack-size</i>	The size of the stack.  Enter a number of machine words that represents the stack size. The following example shows the use of the Itemize environment with its default values in effect.

+ Environment      Creates superscript displays of numbers or letters.

*Markup*

```
(a + b)
^+
 2
^+
= a
^+
+ 2ab + b
^+
 2
^+
.
```

*Display*

```
(a + b)2 = a + 2ab + b2 .
```

- Environment      Creates subscript displays of numbers or letters.

*Markup*

```

H
\mathbb{F}
      2
\mathbb{C}
O is also known as water

```

*Display*

```

H2O is also known as water.

```

**B Environment**      Formats its contents using Facecode B, bold.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

```

This text is in B facecode.

```

**BI Environment**      Formats its contents using Facecode BI, bold italic.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

```

This text is in BI (bold italic) facecode.

```

**Blankpage Command**

Places blank pages into the document. Takes the number of blank pages to be added as a value.

**Blankspace Command**

The Blankspace command leaves vertical blank space in the displayed record.

Use one of these measures to indicate a vertical distance: number of lines, inches, centimeters, pixels, picas or points.

**Box Environment**      Draws a box around the specified text. You can modify the `LeftMargin` and/or `RightMargin` attributes (respectively) to use relative document margins (a signed attribute value), or to use global margin settings (an unsigned attribute value).

This text is in a Box environment.

**C Environment**      Formats its contents using SMALLCAPS.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

THIS TEXT IS IN C (SMALLCAPS) FACECODE.

**Caption Command**      Specifies the caption for a figure or table. The "Tag Command" must follow this command to generate appropriate crossreference labels.

### *Markup*

```

Figure
⌘ Ref, Code horizontal-bug
shows a bug.
Figure ...
    ⌘ Picture "bug-4" from file SCRC\SAP:>sys>
    ⌘ Caption A Bug
    ⌘ Tag, Code horizontal-bug
Figure ...
    
```

### *Display*

Figure 1 shows a bug.

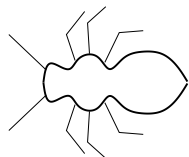


Figure 1. A Bug

### Case Command

Defines Symbolics Concordia document formatting options. Specify a Case selector and zero or more Case clauses.

You can use either the `m-X` "Create Command" or `s-M` (Create Markup) to add Case markup to your document.

Formatting options apply to all of the following document viewing methods:

- "Format Pages Page Previewer Command "
- "Show Documentation Command" (in the Page Previewer, Document Examiner, or the Symbolics Concordia editor)
- `s-P` (in Symbolics Concordia)

Case markup looks like this in your Symbolics Concordia record:

```

|Case
  Selector: LANGUAGE
  |C
    This is C stuff.
    ➔ Include link: Introduction to Using C (section)
  |C
  |Fortran
    This is Fortran stuff.
    ➔ Include link: Introduction to Using FORTRAN (section)
  |Fortran
  |Unassigned
    This is formatted when the Language Case selector is not set to C or
    Fortran.
  |Unassigned
    Click here to add a new clause
|Case

```

You can collapse the view of the Case markup into a single line with `s-sh-Middle`. (Use `s-sh-Middle` again to restore the original view).

#### Case selector

Names a special variable in the sage package. For example,

```
Case selector: sage::language
```

Set the value of the Case selector variable before you format the document to select particular formatting options. For more information, see the section "Setting Document Formatting Options in Symbolics Concordia", page 190.

#### Case clause

Contains documentation that is formatted when the value of the associated Case selector variable matches the name of the Case clause. For example, in the Case environment with the selector **sage::language**, the clause named **FORTRAN** contains text that is formatted when **sage::language** is set to **FORTRAN**.

Note that the names of Case clauses are alphanumeric case insensitive (that is, "Fortran" is equivalent to "FORTRAN" for document formatting purposes).

Case clauses can contain anything that can appear in a Contents field of a Symbolics Concordia record (including other Case commands).

When you format the document, any Case clauses which do not match the value of the Case selector variable are ignored, with the following exceptions:

- Unassigned clauses (if present) are formatted when the selector is not set.
- Otherwise clauses (if present) are formatted when the Case selector is set, but its value does not match any of the other clauses.

Click on items in the expanded view of the Case markup to specify or edit the Case selector value or clauses:

- To replace a selector value in the minibuffer, click **Left** on it.
- To edit a selector value in the minibuffer, click **Middle** on it.

- To add a new clause, click **Left** on   
*Click here to add a new clause.*

Then type the name of the clause in the minibuffer. Note that new clauses are placed before any **Otherwise** or **Unassigned** clauses. Also note that **Case** clauses cannot be copied or reordered.

- To delete a clause, click **≡-Middle** on the clause markup icon.
- To change the name of a **Case** clause, click **≡-Middle** on the markup icon, and type the new name in the minibuffer.

### Center Environment

An "unfilled environment" that causes each line within it to be centered in the displayed record.

#### *Markup*

```
|Center  
  John Q. Public  
  123 Main St.  
  Anytown, U.S.A.  
|Center
```

#### *Display*

```
John Q. Public  
123 Main St.  
Anytown, U.S.A.
```

### Checklist Environment

A filled environment which produces a list marked by pointer icons on the screen. There are no markers when this environment is hardcopied.

Checklist is similar to the **Itemize** environment and **Enumerate** environment.

Using the default attributes, the Checklist environment generates a simple list.

### *Markup*

```
Checklist
  eggs
  bacon
  toast
Checklist
```

### *Display*

```
 eggs
 bacon
 toast
```

The Checklist environment can be altered by changing its default attributes. Refer to "Itemize Environment" to see typical alterations.

### Commentary Environment

Creates an environment for comments. The contents of this environment are not printed or displayed as part of the document.

### CrossRef Environment

Makes its contents mouse-sensitive as a crossreference. You use a CrossRef environment with the invisible view of a crossreference link. The CrossRef environment encloses the text you want to have mouse-sensitive and the invisible crossreference link.

For example, you can use the Crossref command to make a mouse-sensitive link from the text "Select Entity" to the record called "Select Entity Graphic Editor Command". In this example "Select Entity" is a mouse-sensitive invisible link to the record "Select Entity Graphic Editor Command".

*Markup*

```

You can use
<CrossRef>
  Select Entity
  < Invisible link to "Select Entity Graphic Editor Command" Fragment
<CrossRef>
to select entities in your drawing.

```

*Display*

You can use Select Entity to select entities in your drawing.

## Description Environment

Description is a "filled environment" that has a wider left margin after the first line. Because it resembles a two-column table, Description is often used to format lists of definitions. It is also useful for displaying paragraphs of text with headers.

You must insert a tab character to separate the header from the body of the paragraph. Press `s-TAB` or use the Tab-to-tab-stop command.

*Markup*

```

<Description>
  Thing One[→ This is a description of Thing One. You can
  nest Description environments for lists within lists.

  [→ Begin multiple paragraphs for an item with s-Tab or
  enclose all of the paragraphs for the item within a
  Multiple environment.
</Description>

```

*Display*



Thing One	This is a description of Thing One. You can nest Description environments for lists within lists.  Begin multiple paragraphs for an item with <code>\Tab</code> or enclose all of the paragraphs for the item within a Multiple environment.
-----------	--

### Display Environment

Display is an "unfilled environment" where each line in the Concordia file produces one line in the processed text. Both the left and the right margins are widened.

It is often used to show lists without tick-marks or numbers.

This text is in a Display environment. This is a new line.
---

### Enumerate Environment

Enumerate is a "filled environment" that produces a numbered list. It is very similar to the "Itemize Environment" and the "Checklist Environment".

Using the default attributes, the Enumerate environment generates a simple numbered list.

Nested Enumerate environments produce lists that are lettered and numbered, and so are commonly used to make outlines.

*Markup*

```

Enumerate
  Thing One

  Thing Two

  Other Things
Enumerate

```

*Display*

```

1.  Thing One

2.  Thing Two

3.  Other Things

```

**Example Environment**

Example is an "unfilled environment" that creates a region with a wider margin and that uses a fixed-width font. It is designed to show examples of computer interaction.

Often, examples of actual programs contain lines that are too long to be printed on one line in the Example environment. Refer to the section "Longlines Attribute" for more information.

```

(setq b 1)
'(a b c) => (a b c)
'(a ,b c) => (a 1 c)
'(abc ,( + b 4) ,(- b 1) (def ,b)) =>
      (abc 5 0 (def 1))

```

**Figure Environment**

Allows you to specify a caption and a tag for a Figure. Figures are numbered. The caption appears in the List of Figures, generated automatically by the formatter, and the tag can be used with the "Ref Command" for crossreference purposes.

*Markup*

```

Figure
⌘ Ref, Code horizontal-bug
shows a bug.
|Figure ...
  ⌘ Picture "bug-4" from file SCRC\SAP:>sys>
  ⌘ Caption A Bug
  ⌘ Tag, Code horizontal-bug
|Figure ...

```

*Display*

Figure 1 shows a bug.

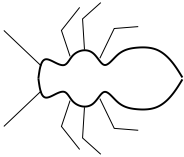


Figure 1. A Bug

**Flushleft Environment**

Flushleft environment is an "unfilled environment" that causes each line within it to be aligned with the displayed record's left "global margin". The distance to the margin can be changed by altering the "Leftmargin Attribute".

This text is in a Flushleft environment.  
This is a new line.



Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

Τηισ τεξτ ισ ιν Γ (Γρεεκ) φαχεχοδε.

**Group Environment** Causes the text within the environment to be grouped together and to appear all on one page. If the grouping is too large to fit on the current page, a new page will be started, leaving white space at the bottom of the previous page. The Group environment should be used only when absolutely necessary. If the grouped text is more than a few lines, it is a good idea to identify one or more points where a page break would be acceptable and insert a Hinge command.

**Heading Environment**

Formats its contents in the same style as a first level heading (chapter) as defined in the book design. This heading is unnumbered.

**This is in a Heading environment.**

**Hinge Command** Indicates the places in a grouped environment where a new page can begin.

**I Environment** Formats its contents using Facecode I, italic.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

*This text is in I (Italics) facecode.*

**Index Command** Makes the text given as the argument to this command an index entry.

**IndexPrimary Command**

Makes the text given as the argument a primary index entry.

### IndexSecondary Command

Makes the text given as the argument a secondary index entry.

### InputExample Environment

An unfilled environment that has a wider margin and uses a fixed-width font. `Inputexample`, `Fileexample`, and `Outputexample` are markups that are identical to `Programexample`.

```
(cp:make-command-table "Local"
 :if-exists :update-options
 :inherit-from '("user")
 :command-table-size 10
 :kbd-accelerator-p nil)
```

### Itemize Environment

`Itemize` is a "filled environment" that produces a bulleted list. The items in the list are marked by bullets, or tick-marks. Separate each item by one blank line.

The `Enumerate` and `Checklist` environments are similar to `Itemize`.

#### *Markup*

```
Itemize
  Thing One

  Thing Two

  Other Things
Itemize
```

#### *Display*

- Thing One
- Thing Two
- Thing Three

**K Environment** Formats its contents using Facecode K, a character set that looks like computer keyboard input.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

```
This text is in K (Keyboard) facecode.
```

**Label Environment** Defines a crossreference label.

*Markup*

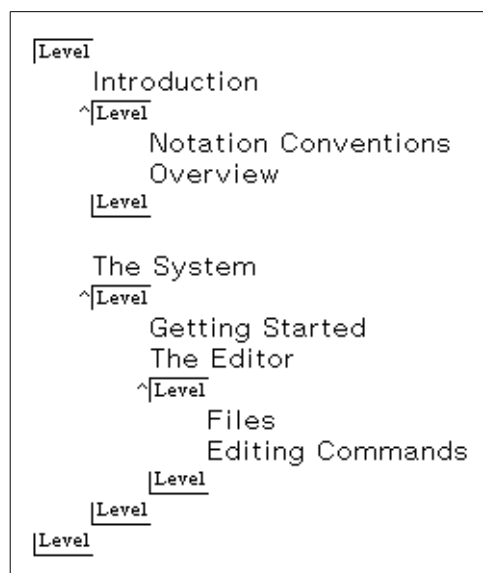
```
☞Label, Code George
This is section
☞Ref, Code George
in the document.
```

*Display*

```
This is section 1 in the document.
```

**Level Environment** An environment for writing outlines. Level is similar to Enumerate, except that it uses the standard conventions for numbering outlines.

*Markup*



### *Display*

- I. Introduction
  - A. Notation Conventions
  - B. Overview
- II. The System
  - A. Getting Started
  - B. The Editor
    - 1. Files
    - 2. Editing Commands

LS Environment    Formats its contents using Facecode LS, the same style as Lisp Objects.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

**This text is in LS (Lisp object) facecode.**



### MajorHeading Environment

Formats its contents in the same style as a Majorpart as defined by the book design. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup.

**This is in a MajorHeading environment.**

**Modify Command** Changes attribute values throughout a document for a particular environment.

You can use this command in top-level records to implement local formatting conventions for the document.

### Multiple Environment

Used inside Itemize, Enumerate, and Description environments to indicate that a series of paragraphs are all part of one item for purposes of formatting.

### NewPage Markup Command

Begins a new page immediately. Its argument is an integer that controls the number of blank pages to leave (the default is 0).

### OutputExample Environment

An unfilled environment that has a wider margin and uses a fixed-width font. Outputexample, Fileexample, and Inputexample are markups that are identical to Programexample.

This text is in an OutputExample environment. Note the wider margins.

**P Environment** Formats its contents using Facecode P, bold italic.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

*This text is in P (bold italic) facecode.*

### PageFooting Command

Selects the text and format for the running text at the foot of a page or pages.

### PageHeading Command

Selects the text and format for the running text at the top of a page or pages.

**PageRef Command** Use with the **Tag** command for inserting page references.

### PermanentString Command

Used with the **Value** command to set document formatting options (that is, Case selector variable values) that persist across formatting runs. (String assignments may change when the formatter finishes). Use the **Value** command to associate the permanent string with a value.

For more information, see the section "Setting Document Formatting Options in Symbolics Concordia", page 190.

### Presentation Environment

Allows you to specify a **Presentation-Type**, **Presentation-Object**, and **Presentation-Options** for arbitrary mouse sensitivity. See the section "Creating Mouse Sensitivity in Your Output", page 469.

### ProgramExample Environment

**ProgramExample** is an unfilled environment which creates a region with a wider margin and which uses a fixed-width font. It is similar to the environments "Example Environment" and "Display Environment". It is designed to show examples of computer programs.

```
(defmethod (screen-arrow-output :show-lines)
  (x y &rest x-y-pairs)
  (loop for x0 = (send self ':compute-x x) then x1
        for y0 = (send self ':compute-y y) then y1
        for (x1 y1) on x-y-pairs by #'cddr
        do (setq x1 (send self ':compute-x x1)
              y1 (send self ':compute-y y1))
          (send terminal-io ':draw-line
                x0 y0 x1 y1 tv:alu-i-or t)))
```

**Quotation Environment**

Formats its contents in a filled environment with left and right margins increased by one centimeter.

It is not the fashion to see the lady the epilogue;  
but it is no more unhandsome than to see the lord  
the prologue. If it be true that good wine needs no  
bush, 'tis true that a good play needs no epilogue.  
Yet to good wine they do use good bushes, and  
good plays prove the better by the help of good  
epilogues. - *As You Like It*.

**R Environment**

Formats its contents using Facecode R, roman.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

This text is in R (Roman) facecode.

**Ref Command**

Retrieves and prints the value associated with a counter used to count page numbers, figures, tables, chapters, sections, and appendixes.

*Markup*

☞ Label, Code George

This is section

☞ Ref, Code George  
in the document.

*Display*

This is Section George in the document.

**S Environment** Formats its contents using Facecode S, usually a symbol character set.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

```
™ηισ τεξτ ισ ιν Σ (Σψ—β•λ) φαχεχ•δε°
```

**Set Command** Sets a counter to a value. You specify the counter as a keyword and then specify the value.

**SimpleTable Environment**

Formats its contents as a simple table. Enter the contents of the table with columns separated by tabs. Concordia calculates the tabstops for you based on the length of the entries. See the section "Creating Simple Tables", page 135.

*Markup*

```
Simpletable
☞ SimpleTableSpecs Hrules 1; Vrules 1,2
  thing described[→ its discussibility[→ what it means
  :this[→ at hand[→ what this means
  :that[→ remote[→ What that means
Simpletable
```

*Display*

<i>thing described</i>	<i>its discussibility</i>	<i>what it means</i>
<b>:this</b>	at hand	what this means
<b>:that</b>	remote	What that means

Empty cells are ignored, so if you want a gap in your table (say, if you wanted "at hand" above to be a blank), you would place a space between the two tabs in the input.

You can omit the `SimpleTableSpecs` and have no rules. However, if you supply `SimpleTableSpecs` you must supply at least one vertical and one horizontal rule. You do not have to specify a continuous series of rules. For example, if you have an item in your table that runs onto two lines, you can skip that position in the horizontal rules so that both lines of the item end up in the same "box". This `SimpleTable` has horizontal rules at positions 1, 2, and 5:

This	Here
That	There
The Other Thing That Takes More Than One Line	Way Over Yonder
Another One	Nearby

#### `SimpleTableSpecs` Environment

Specifies horizontal and vertical rules for a simple table. See the section "Creating Simple Tables", page 135.

`String Command` Defines a text string to be used with the `Value` command.

You can use this command to set the specified `Case` selector variable to an ambient value. For more information, see the section "Setting Document Formatting Options in Symbolics Concordia", page 190.

`Style Command` Specifies formatting style directives, for example `Justification`.

#### `SubHeading` Environment

Formats its contents as a subheading, that is in the same style as a third level heading (subsection) is formatted as defined in the book design. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup.

**This is in a `SubHeading` environment.**

#### `SubSubHeading` Environment

Formats its contents as a subsubheading, that is in the same style as a fourth level heading (subsubsection) is formatted as

defined by the book design. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup.

***This is in a SubSubHeading environment.***

**T Environment**      Formats its contents using Facecode T, a fixed width (typewriter) character set.

Note that creating this environment without a marked region changes the typein character style. (You can also use `c-m-J` to change the typein character style.)

This text is in T (Typewriter) facecode.

**TabClear Command** Eliminates any tabs that were previously set in a file. Usually used before any type of tab setting command, like "TabDivide Command" or "TabSet Command".

**TabDivide Command**

Divides the formatting region into columns. Columns are of equal widths and sizes to fill the region horizontally. Text items are separated for formatting by using the "`␣-TAB` Command" or by pressing `␣-TAB`.

This command takes an integer as an argument.

Although it is not necessary to clear tabstops within an environment, doing so guarantees that the TabDivide command will work as you expect it to. It is a good work habit to put a TabClear command before each TabDivide command you use.

When equally sized columns are not desired, use the "TabSet Command".

Wrap tabbed text in a Format Environment to override any formatting defaults.

This is an example of how to set up a two-column table.

*Markup*

*Wrappers* ↔ *Whoppers*

Similar to a macro. ↔ Similar to a function  
 If a wrapper is modified, all ↔ If a whopper is modified,  
 Combined methods using it must ↔ only the whopper must  
 be recompiled (this is done ↔ recompiled.  
 automatically.)

*Display**Wrappers*

Similar to a macro.  
 If a wrapper is modified, all  
 Combined methods using it must  
 only the whopper must be  
 be recompiled (this is done  
 automatically.)

*Whoppers*

Similar to a function  
 If a whopper is modified,  
 recompiled.

This is an example of how to set up a three column table.

*Markup*

☞ TabDivide 3

*Character Style* ↔ *Default* ↔ *Result of*  
*of a Character* ↔ *Character Style* ↔ *Merging*

NIL.NIL.NIL ↔ FIX.ROMAN.NORMAL ↔ FIX.ROMAN.NORMAL  
 NIL.ITALIC.LARGE ↔ FIX.ROMAN.NORMAL ↔ FIX.ITALIC.LARG  
 NIL.ITALIC.SMALLER ↔ FIX.ROMAN.NORMAL ↔ FIX.ITALIC.SM  
 SWISS.BOLD.LARGER ↔ FIX.ROMAN.NORMAL ↔ SWISS.BOLD.L

*Display*

<i>Character Style of a Character</i>	<i>Default Character Style</i>	<i>Result of Merging</i>
NIL.NIL.NIL	FIX.ROMAN.NORMAL	FIX.ROMAN.NORMAL
NIL.ITALIC.LARGE	FIX.ROMAN.NORMAL	FIX.ITALIC.LARGE
NIL.ITALIC.SMALLER	FIX.ROMAN.NORMAL	FIX.ITALIC.SMALL
SWISS.BOLD.LARGER	FIX.ROMAN.NORMAL	SWISS.BOLD.LARGE

**Table Environment** Allows you to specify a caption and a tag for a Table. Tables are numbered. The caption appears in the List of Tables, generated automatically by the formatter, and the tag can be used with the "Ref Command" for crossreference purposes.

**TabSet Command** Creates tabs within the text at distances you specify.

- Use a horizontal distance: number of characters, inches, or centimeters.
- Use the "s-TAB Command" or s-TAB to separate text items.
- Insert "TabClear Command" to clear previously set tabs.

It is a good work habit to put a TabClear command at the end of any environment where tabs are explicitly set. Clearing tabs that are no longer necessary, guarantees that the TabSet command will work as you expect it to.

Wrap tabbed text in a Format Environment to override any formatting defaults in the active environment.

*Markup*

```

Format
  ↵ TabClear
  ↵ Tabset, Tabs 1.5 Inches, 3.25 Inches, and 4 Inches
  array[↵ vector[↵ list[↵ symbol
  simple-array[↵ bit-vector[↵ cons[↵ null
  simple-vector[↵ simple-bit-vector[↵ [↵ keyword
  structure
Format
...

```

Notice that a column may be left blank by inserting two tab icons in a row.



*Display*

array	vector	list	symbol
simple-array	bit-vector	cons	null
simple-vector	simple-bit-vector		keyword
structure			

**Tag Command** Specifies a *Codeword* (a string) as a crossreference label showing the position and number of an equation, theorem, figure, or table.

Used with the "Caption Command" to mark figures and tables.

Referenced by the "Ref Command".

**Text Environment** It may seem strange to have a special environment called Text, because after all everything we type that is not in a special environment is processed just like stuff written in the text environment.

This paragraph is written in the Text in environment.

The reason the Text environment is included is for writers who want to change the default attributes of the text. For instance, the writer decided this paragraph should have a 2 inch left margin, so she changed the Leftmargin attribute on the c-m-Right menu to 2 inches.

For this paragraph, she decided the right margin should be wider.

And here, she decided that both the margins should be narrower. Thus she inserted a value of -0.5 inches as the attribute value for both Leftmargin and Rightmargin.

Note that you can use plus (+) or minus (-) signs to define Leftmargin or Rightmargin settings relative to the current global margin settings. Unsigned attribute values specify absolute margins.

**TitleRef Command** Retrieves and prints the value associated with a counter, in the titlestyle using the titlefont. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup and references are done using crossreference links.

**Transparent Environment**

An environment that has no effect. It exists to permit you to adjust an attribute for a small part of a document without altering the design of the document as a whole. For example, you can scale a picture that is just inserted into text (not in a figure or example environment) by adding the PictureScale attribute to a Transparent environment that you have wrapped around the picture.

**U Environment** Underlines the non-blank characters (including punctuation) in its contents.

This text is in U (Underline) environment.

**UN Environment** Underlines the alpha-numeric characters in its contents.

This text is in UN (Underline alpha-numeric) environment.

**UnNumbered Command**

Modifies environments that have the Numbered attribute so that paragraphs are unnumbered.

**Use Command** Takes an environment name as its argument. This definition the environment is included as part of the current environment.

**UX Environment** Underlines its contents, including spaces and punctuation.

This text is in UX (Underline all) environment.

**Value Command** Retrieves and prints the value of its argument, which should be a counter or a string defined by the String command.

You can also use the Value command to retrieve the value of "Sage System Variables".

### Verbatim Environment

Verbatim produces everything you type exactly as you type it, without moving margins or justifying text. It is used to show unusual formatting.

Verbatim is an "unfilled environment" like "Format Environment" except that Verbatim uses a fixed-width font unlike regular text.

```
See
    how
        you
            can
                position
                    text
                        anywhere.
```

You can also use Verbatim to make tables and figures.

```

      car-----cdr
      |
      car-cdr      car-----cdr
      | |          |
MYTILUS MUSSEL   car-----cdr      car---cdr
                  |                |
                  WHELK             car-cdr nil
                  |                |
                  car-----cdr     | |
                  |                 | 7 4
                  PERIWINKLE       |
                  |
                  car-----cdr
                  |                |
                  car-cdr          car-cdr
                  | |              | |
FAMILIES 5     SHELLS nil
```

**Verse Environment** An filled environment for printing verse. Each line is treated as a paragraph, that is each line of verse is terminated by RETURN and if a line is too long for the linelength, it wraps with appropriate indentation on the continuation line.

Tw'as brillig, and the slithy toves  
 Did gyre and gimble in the wabe.  
 All mimsy were the borogoves,  
 And the mome raths outgrabe.  
*-Lewis Carroll*

**Word Environment** Indicates that its contents are to be treated as a single word and should not be hyphenated.

### 15.2.6. List of Symbolics Concordia Attributes

<b>Above Attribute</b>	Determines the amount of white space preceding the environment.
<b>Below Attribute</b>	Determines the amount of white space following the environment.
<b>Blanklines Attribute</b>	Determines the treatment of blank lines within the environment.
<b>Bottommargin Attribute</b>	Determines the vertical distance between the last line of text and the end of the page.
<b>Boxbm Attribute</b>	Determines the bottom margin of a box specified with the boxed environment. Takes a value that is the distance between the last line of boxed text and the box outline.
<b>BoxFlushRight Attribute</b>	Ses the right margin of a box to be flush with the right-hand margin for that page.
<b>Boxlm Attribute</b>	Determines the left-hand margin of a box specified with the boxed environment. Takes a value that is the distance between the left margin of the boxed text and the box outline.
<b>Boxrm Attribute</b>	Determines the right-hand margin of a box specified with the boxed environment. Takes a value that is the distance between the right margin of the boxed text and the box outline.
<b>Boxtm Attribute</b>	Determines the top margin of a box specified with the boxed environment. Takes a value that is the distance between the box outline and the first line of boxed text.

- Boxtype Attribute** Controls whether the contents of an environment is surrounded by a box.
- Break Attribute** Sets the line breaks surrounding an environment.
- Capitalized Attribute**  
Controls the capitalization of text. When you specify a value of true, all text appears in upper case. When you specify a value of false, text is left as is. The default value is false.
- Centered Attribute** Centers text between the left and right margins.
- Columnmargin Attribute**  
Sets the space between columns of text.
- Columns Attribute** Sets the number of columns that text is printed in.
- Columnwidth Attribute**  
Sets the width of a column. Its value includes the width of the text and the width of the column margin.
- Continue Attribute** Used with the Break attribute to determine whether the text following the environment begins a new paragraph or continues the old one.
- CRBreak Attribute** Determines whether carriage returns result in a paragraph break.
- CRSpace Attribute** Causes a carriage return to be treated as a space character.
- FaceCode Attribute** Selects a facecode.
- Fill Attribute** Causes lines of text to be filled, that is text runs from the left margin to the right margin. A line of text can have one of the attributes: Fill, Centered, FlushRight, or FlushLeft.
- Fixed Attribute** Specifies the place on a page where the text in this environment appears. Takes a vertical distance as a value. Use one of these measures to indicate a vertical distance: number of lines, inches, centimeters, pixels, picas or points.
- Float Attribute** Causes the text within the environment to appear all on one page. If the grouping is too large to fit on the current page, the current page is filled and the "floated" text appears on the next new page. An environment can use one of the attributes: Float, FloatPage, Free, or Group.
- FloatPage Attribute** Causes the text within the environment to appear on the next whole page. The current page is filled and the "floated" text appears on the next new page. An environment can use one of the attributes: Float, FloatPage, Free, or Group.
- Flushleft Attribute** Causes each line within the environment to be aligned with the record's left "global margin". The distance to the margin can be changed by altering the "Leftmargin Attribute".

**FlushRight Attribute**

Causes each line within the environment to be aligned with the record's right "global margin". The distance to the margin can be changed by altering the "Rightmargin Attribute".

**Font Attribute**

Selects a font.

**Free Attribute**

Removes any grouping restrictions such as Float or Group from the text in an environment. An environment can use one of the attributes: Float, FloatPage, Free, or Group.

**Group Attribute**

Causes the text within the environment to be grouped together and to appear all on one page. If the grouping is too large to fit on the current page, a new page will be started, leaving white space at the bottom of the previous page. An environment can use one of the attributes: Float, FloatPage, Free, or Group.

**Hyphenbreak Attribute**

Controls whether a hyphen in text can be treated as a line break, if needed. Takes a boolean value.

**Increment Attribute**

Increments a counter each time the environment specified as the value for this attribute is invoked.

**Indent Attribute**

Controls the displacement of the first line of each paragraph in the environment from the left margin specified in the document type.

**Indentation Attribute**

Controls the displacement of the first line of each paragraph in the environment from the left margin specified in the document type. This attribute is identical to the Indent attribute.

**Justification Attribute**

Aligns text with the prevailing right-hand margin. Text justification works only in filled environments.

**LeadingSpaces Attribute**

Determines how leading spaces are treated in this environment.

**Leftmargin Attribute**

Sets the width of the left margin.

**Linewidth Attribute**

Sets the maximum width for a line of text.

**Longlines Attribute**

Specifies what happens to lines that would extend past the right margin of the current environment. Valid only for unfilled environments.

**NarrowestBlank Attribute**

Defines a distance representing the minimum size the horizontal justification routine will allow a space to be shrunk to.

Need Attribute	Checks that the requested amount of space exists on the page. If not, begins a new page. Takes a vertical distance as a value.
Numbered Attribute	Selects a format for numbering paragraphs.
Numberfrom Attribute	Provides the initial value for numbering paragraphs in an environment.
NumberLocation Attribute	Specifies where paragraph numbers appear.
Pagebreak Attribute	Specifies whether a page break must precede or follow an environment.
Paragraphbreaks Attribute	Controls whether a paragraph break increments a counter, accepting an argument of Normal or Limited.
Picturescale Attribute	A number, a relative number (as in *2), or a vertical distance that controls how to scale any pictures contained within the environment.
Presentationobject Attribute	Allows arbitrary presentations in documents. The value of this attribute should be a form that, when evaluated, produces the object of the presentation.
Presentationtype Attribute	Allows arbitrary presentations in documents. The value of this attribute should be a form that, when evaluated, produces the presentation type of the presentation.
Referenced Attribute	Selects a format for references to numbered paragraphs.
Rightmargin Attribute	Sets the width of the right margin.
Script Attribute	Positions the baseline of the environment in relation to the usual text baseline.
Sink Attribute	Determines the amount of white space between the environment and the top margin for text.
Size Attribute	When used in conjunction with <b>Font</b> , specifies the size for the font, in points.
Spaces Attribute	Controls how spaces are treated in this environment.
Spacing Attribute	Specifies baseline-to-baseline vertical spacing.

Spread Attribute	Specifies the amount of space added to the Spacing attribute value to determine the vertical space between paragraphs.
TabExport Attribute	Controls whether tab settings for an environment are cleared following the environment.
TopMargin Attribute	Controls the vertical distance between the top of the page and the first line of text.
Underline Attribute	Controls what characters are underlined.
UnNumbered Attribute	Modifies environments that have the Numbered attribute so that paragraphs are unnumbered.
Use Attribute	Takes an environment name as its argument. This definition of this environment is included as part of the current environment.
Widestblank Attribute	A distance representing the maximum size the horizontal justification routine will allow a space to be stretched to.
Within Attribute	Used for a template defined with the Numbered attribute, the Within attribute creates a parent counter.

### 15.3. Character Styles in Symbolics Concordia

You can create emphasis or enhance readability by using different typefaces in their text. In Symbolics Concordia, you can change a typeface by changing the *character style*. The Concordia editor shows a semblance of what the hardcopy document will look like: Bold, italic, and fixed-width faces appear as such, rather than being indicated by font change characters or embedded notations. Character styles work the same as in the rest of Genera. Only a brief description is given here of the four basic commands. For more information, see "Understanding Character Styles" and "Using Character Styles in Zmacs".

<i>Keystroke</i>	<i>Meaning</i>
c-m-J	Changes the <i>typein</i> style.
c-X c-J	Changes the style of a <i>region</i> of existing text.
m-J	Changes the style of the <i>word</i> following the cursor.
c-J	Changes the style of the <i>character</i> following the cursor.

Abbreviations for valid character styles are:



Abbreviation	Character Style
I	<i>italic</i>
B	<b>bold</b>
K	keystroke (Note: It does not coerce uppercase.)
T	typewriter
B-I	<b><i>bold-italic</i></b>
L	<b>lisp-like</b> (Note: It does not coerce lowercase.)

`c-m-J` changes the typein style. New text will appear in the character style you select.

Example: Typing `c-m-J B RETURN` puts new typein in boldface.

Use `c-m-J` to reset the typein style to the default regular style. This command is also available as Change Typein Style under **Character Style** in the menu.

`c-X c-J` changes the style of a region of existing text. Example: Mark a region and type `c-X c-J I RETURN`. The marked text is put in an italics. This feature is also available from the menu. See the following commands under **Character Style**:

- "Change Region to Italic",
- "Change Region to Bold", and
- "Change Region to Regular".

`m-J` changes the style of the word following the cursor. For example, `m-J I RETURN` places the next word in italics. You can type `m-J` successively to continue changing the character style of the next word, without needing to reselect the character style. For example, `m-J I RETURN m-J m-J` renders the three words following the cursor in italics. Alternatively, typing `m-3 m-J I RETURN` has the same effect.

`c-J` changes the style of the character following the cursor. It operates the same as `m-J`.

Note that you can use the `m-X Show Character Styles` command to display the character styles used in a region of text or in a buffer.

## 15.4. Marking up Lisp Language Objects

Lisp language objects should be placed in language markup, which makes them mouse-sensitive in Document Examiner. Clicking Left on the name of a Lisp object in Document Examiner reads in its documentation record. In effect, you are inserting an implicit crossreference to the documentation for this Lisp object.

### Make Language Form

Inserts markup for the name of a Lisp object. You type the

name of the object and invoke the command; the object name is placed in a boldface character style and enclosed by two small glyphs. In Document Examiner the name of a Lisp object is mouse-sensitive. A Lisp object in Language markup is an implicit link to the record (of the same name) documenting the Lisp object. Clicking on a Lisp object displays its documentation.

This command is available as **Make Language Form** under **Markup** in the menu or as `⌘-L`.

## 16. Moving Around Symbolics Concordia

### 16.1. Moving Around the Symbolics Concordia Buffer

In addition to the standard scrolling and cursor movement commands, specific commands are provided for Symbolics Concordia buffers.

#### Locating the Boundaries of a Record

##### Beginning of Record

Moves the cursor to the header of the current record. Executing the command again moves the cursor to the header of the immediately preceding record.

This command is available as Beginning of Record under **Records** in the menu and as `s-c-A`.

##### End of Record

Moves the cursor to the trailer of the current record. Executing the command again moves the cursor to the trailer of the next record.

This command is available as End of Record under **Records** in the menu and as `s-c-E`.

#### Locating the Boundaries of a Record Field or Environment

These commands operate within a single record.

- Beginning of Environment
- End of Environment

These commands operate on the records in a buffer.

- Find Markup
- Reverse Find Markup

##### Beginning of Environment

Moves the cursor to the beginning of the nearest environment or record field.

This command is available as Beginning under **Markup** in the menu or as `s-(`.

##### End of Environment

Moves the cursor to the end of the nearest environment or record field.

This command is available as **End** under **Markup** in the menu or as  $\varepsilon-$ ).

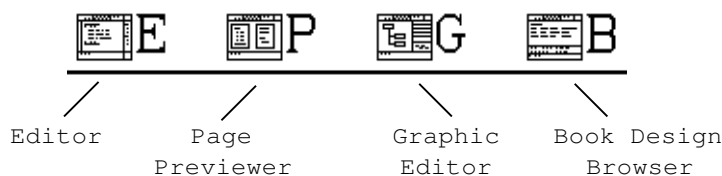
**Find Markup** Prompts for a markup environment (for example, Commentary, Figure, or Enumerate). It then searches for a markup environment of that type. If it finds one, it positions the cursor on the environment diagram line for the environment. Both the beginning and ending environment diagram lines are found by this command.

**Reverse Find Markup** Prompts for a markup environment (for example, Commentary, Figure, Enumerate). It then searches for a markup environment of that type. If it finds one, it positions the cursor on the environment diagram line for the environment. Both the beginning and ending environment diagram lines are found by this command.

## 16.2. Moving Among the Symbolics Concordia Editor, Page Previewer, Graphic Editor, and Book Design Browser

There are several ways to move back and forth among the editor, previewer, graphic editor, and book design browser:

- When you have inserted a picture in a record and you need to return to the graphic editor to edit the picture: Click  $m$ -Left on the representation of the image in the record (looks like a link). When you finish editing your picture, click on Done to return to the Symbolics Concordia editor.
- Press  $\varepsilon$ -SELECT and then type **E** for Symbolics Concordia editor, **P** for Page Previewer, **G** for Graphic editor, or **B** for Book Design Browser.
- Click on the appropriate icon in the Icon pane.



## 17. Inserting Illustrations in Your Symbolics Concordia Documents

The Graphic Editor allows you to insert both illustrations you draw yourself, and full or partial screen dumps, into your document.

**Note:** When you have inserted a picture in a record and you need to return to the Graphic Editor to edit the picture: Click  $\mathcal{M}$ -Left on the representation of the image in the record. When you finish editing your picture, click on Done to return to the Symbolics Concordia editor.

### 17.1. Putting Pictures in Text

You insert a picture using the  $\mathcal{M}$ -X Create Picture Command.

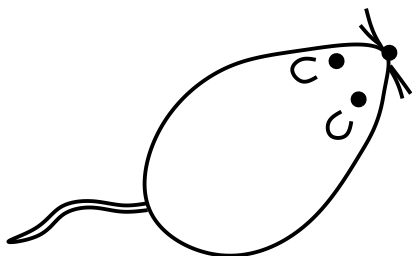
Create Picture      Creates a *representation* of a picture in a Symbolics Concordia file.

You cannot edit the picture in a Symbolics Concordia editor pane. To edit the picture, you invoke the Graphic Editor by clicking  $\mathcal{M}$ -Left on the image in the Symbolics Concordia editor pane. This reads the file containing the source for the picture into the Graphic Editor and selects the picture as the current drawing. When you have edited the picture to your satisfaction, click on [Done] in the menu to return to the Symbolics Concordia editor.

**Note:** Display of pictures on the screen, or formatting them for a printer, is slow. And the more complicated the picture, the slower the display.

A Picture, even a full screen dump, is treated by the formatter as a single character. So pictures can be formatted just the same as text. If a picture is just placed in text, the formatter attempts to fill the line around it just as if it were a large single character.

Here is a picture of a mouse, just inserted in the buffer with paragraph breaks around it:



Formatting environments can be used to position the picture.

Here is the mouse inserted as a Figure (with a Center environment to position it nicely):

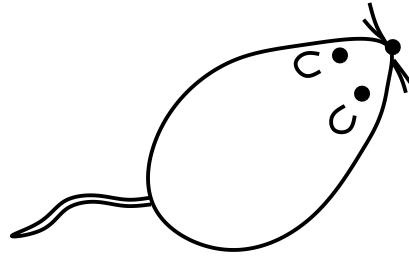


Figure 42. Mouse

For more information, see the section "Using Screen Images as Illustrations", page 322.

### 17.1.1. Specifying the Size of Pictures

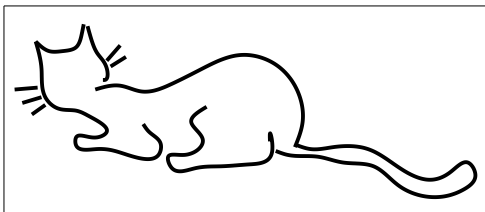
You can scale pictures in three ways:

1. By an absolute factor.
2. Relative to a prevailing factor.
3. To an absolute measure.

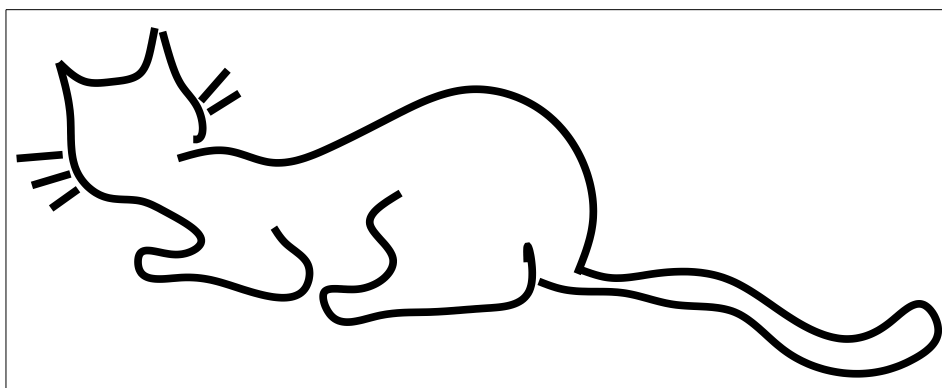
To scale a picture, place it in an environment such as `Figure` and give the environment the additional attribute `PictureScale`. (If you do not want your picture to have any environment attributes other than `PictureScale`, use the environment `Transparent`.)

`PictureScale` Specifies the scaling factor for a picture inserted in a document. Scale can be specified by a factor, such as 2 or .5, relative to a prevailing factor, such as \*2 or \*.5, or in absolute measurements, such as 3 inches.

Scaling by an absolute factor, such as 2, means making the picture four times as large. Here is a mouse-sensitive item scaled by 1 (full size):



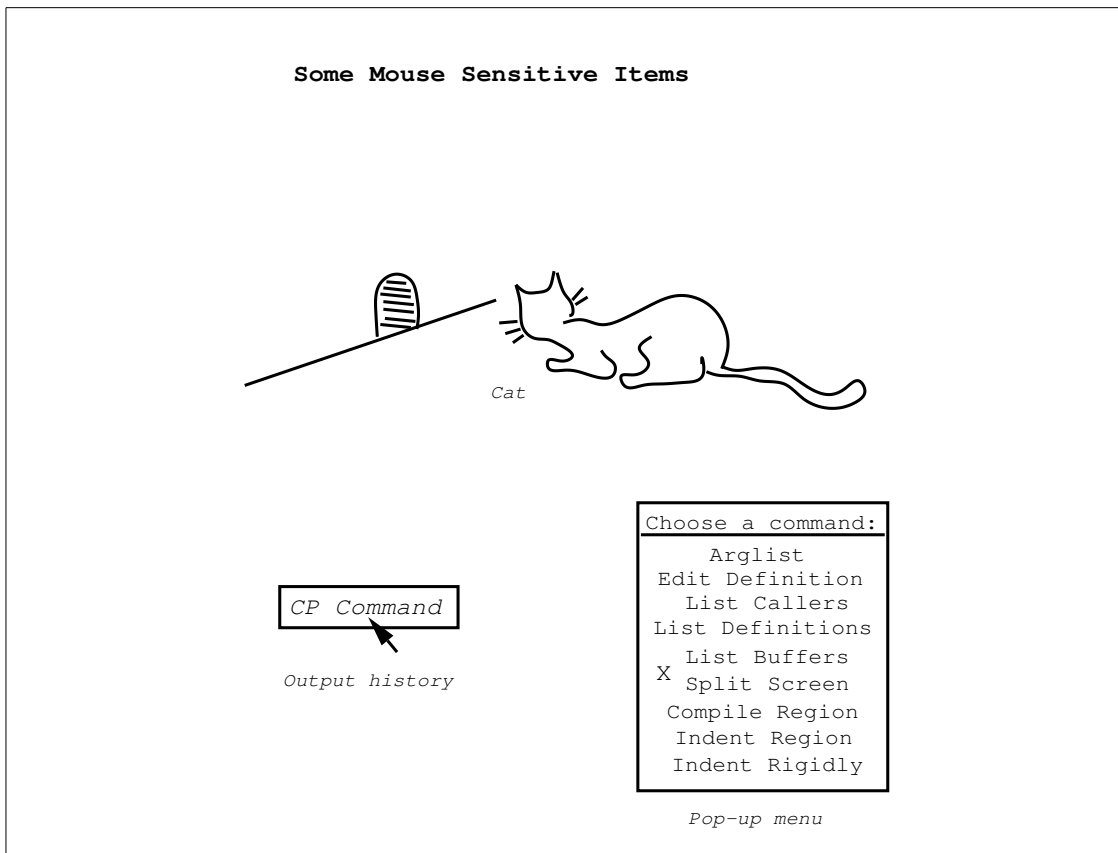
And here it is again, scaled by a factor of 2:



Scaling relative to a prevailing factor means that the scale of all pictures in a document can be kept synchronized. If they are all given relative scales, the overall scale can be changed and all pictures will be rescaled, keeping the same size relationships among themselves. The default prevailing scale is 1, meaning full size as drawn, so scaling something \*2 in that case is equivalent to scaling it by 2. However, if the prevailing scale is .5, a picture scaled by 2 would still display as four times as large, while one scaled \*2 would display as if scaled by 1.

To set the prevailing scale for a document or section of a document, you would either specify it in the book design or wrap a Transparent environment around the contents of the top level record and set the PictureScale attribute of that environment.

Scaling to an absolute measure means you specify the vertical size you want the picture to fill, in inches, centimeters, or pixels. Here is the entire Mouse Sensitive Items picture, scaled to 4 inches:



### Notes about Scaling

Screen display of scaled pictures in the Page Previewer does not always look right. The picture is positioned correctly and takes up the appropriate portion of the page for layout purposes, but due to the small character-style used in the Page Previewer, the elements of the picture are improperly scaled. When you print the formatted pages, the scaled pictures appear correctly.

Display of scaled pictures takes longer than display of regular pictures.

#### 17.1.2. Tips and Techniques for Using Pictures in Documents

This section describes tips and techniques for using pictures (that is, Graphic Editor drawings) in Symbolics Concordia documents.

- Because a representation of the picture is put *in* the record when you use Create Picture, just editing the picture source file does not update what appears in the editor buffer (or for that matter what will appear in the document). You must request the editor to edit the picture, using the Graphic editor (by clicking *m-Left* on the image) and then tell the editor you are done by clicking on [Done] in the Graphic editor.



- Occasionally, when you click on an image to edit it, your Graphic editor comes up blank. If this happens, just click on [Done] and repeat clicking on the image. The picture is then read in properly.
- If you use the same picture in several places in documentation, you probably want to place the picture in a record by itself, then at each point where you want this picture to appear, use a Contents link to the record containing the picture. This way you only have one copy of the picture to update and since pictures do add to the size of files, you save some small amount of disk space.
- When you save your document file, the editor does not advise you that you have a picture file that also needs to be saved, so you must remember to save your picture source files yourself. (Note: if you forget to save the source file, the document file still contains the edited version of the picture, the source is just not changed. This is because what is in the document is *an image of the picture*.)

## 17.2. Creating Active Examples

Active examples are pieces of Lisp code that are run when the documentation is displayed and produce output. They are mouse sensitive and can be run or their code edited by readers from Document Examiner.

To create an active example, use the following steps:

1. Create an ActiveExample markup.
2. Type the code that belongs in the example.
3. Use `m-x Create Example Record Marker` at the end of the code.
4. Use `m-x Test Example` to run the example and save the results.

An example may produce text output, graphics output, and return values.

`m-x Create Example Record Marker` inserts a *record marker* in the example. The record marker saves the results of running the preceding Lisp form. The command prompts you for the type of record marker to create.

You can create the following types of record markers:

Typescript	Saves characters output to <b>*standard-output*</b> .
Picture	Saves the graphics output to <b>*standard-output*</b> . This does not capture text, unless it's drawn with <b>graphics:draw-string</b> .
Values	Saves the values returned by the forms in the example.
Bitmap	Saves the values as a bitmap.

m-X Test Example runs an active example and saves the results of running the example in the record markers embedded in the example.

**Markup:**

```

ActiveExample
(print "FOO BAR BAZ")      ; the output is:

⊕ Example record typescript

(graphics:with-room-for-graphics (t 100)
 (graphics:draw-arrow 20 20 70 70 :thickness 2))

;;; The graphics output is:

⊕ Example record picture

(+ 20 30)                  ; the value is:

⊕ Example record values
ActiveExample

```

**Display:**

```


(print "FOO BAR BAZ")      ; the output is:

"FOO BAR BAZ"

(graphics:with-room-for-graphics (t 100)
 (graphics:draw-arrow 20 20 70 70 :thickness 2))

;;; The graphics output is:

```



```

(+ 20 30)                  ; the value is:

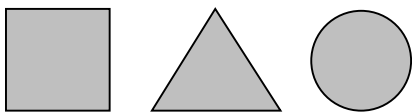
50

```

### 17.3. Making Pictures Using Lisp

Inserts the Lisp code that draws a drawing into an editor buffer. It prompts you for a drawing loaded into the Graphic Editor.

You can then edit the code, or locate and modify specific entities. Here is a simple drawing:



Using Insert Graphic Editor Drawing Code (m-x) inserts the following in your editor buffer:

```
;;; Drawing insert-drawing-code from SAP:>sys>graphic-editor>insert-drawing-code-example.pic

(DEFUN DRAW-INSERT-DRAWING-CODE (&OPTIONAL (*STANDARD-OUTPUT* *STANDARD-OUTPUT*))
  (GRAPHICS:WITH-SCAN-CONVERSION-MODE (T :SKETCH NIL)
    ;; Rectangle-6
    (PROGN (GRAPHICS:DRAW-RECTANGLE 158 438 212 385 :GRAY-LEVEL 0.25)
      (GRAPHICS:DRAW-RECTANGLE 158 438 212 385 :FILLED NIL))
    ;; Triangle-3
    (PROGN (GRAPHICS:DRAW-TRIANGLE 267 438 234 385 301 385 :GRAY-LEVEL 0.25)
      (GRAPHICS:DRAW-TRIANGLE 267 438 234 385 301 385 :FILLED NIL))
    ;; Circle-2
    (PROGN (GRAPHICS:DRAW-CIRCLE 343 411 26 :GRAY-LEVEL 0.25)
      (GRAPHICS:DRAW-CIRCLE 343 411 26 :FILLED NIL))))
```

Presentations can be added to entities, to create mouse sensitivity in parts of the picture. See the section "The Presentation Type System: an Overview" in *Programming the User Interface*.

The modified drawing should be saved in a Lisp file and can then be inserted into a Symbolics Concordia buffer using the Create Picture command and the picture type Lisp.



## 18. Creating an Index in a Symbolics Concordia Document

To create an index, add index entries to the Keywords field. An index entry is one or more words such as "fields" or "record fields" or "fields of a record". Each entry must be on its own line; press RETURN to end an entry.

Your capitalization is retained: "Edit Record command" is indexed the way you typed it. Certain words in an entry are suppressed in the printed index; these include articles (such as "the") and certain prepositions (such as "of").

Each word of an index entry is called a *keyword*; for example, "record" and "fields" are each keywords in the entry "record fields". Each word in the record name is also considered a keyword. For example, this record, "Creating an Index in a Symbolics Concordia Document", contributes seven keywords.

Symbolics Concordia uses the keywords in the following ways:

- They are displayed in the topic overview in Document Examiner.
- They are printed in the index of a hardcopy document.
- Document Examiner uses the keywords to process reader-lookup requests.

Choose keywords that you think a reader might use to look up information.

Since each word in the record name is a keyword, you should not index the record name. The record name automatically appears in the printed index.

### Index Management

There are several Symbolics Concordia editor commands to help with index management. Using these commands can help you in assessing what index entries get generated and where they are coming from. They are useful for identifying index entries that are inconsistent and for locating the records that generated them. You can also use the output from these commands to locate the source of stray index entries that you would like to eliminate. These commands are:

- "Collect Index Entries of Buffer"
- "Collect Index Entries of Tag Table"
- "Collect Index Entries of Topic"

They collect index entries that the formatter would generate for the current buffer, specified tag table, or specified topic respectively. The alphabetized index entries are placed in a new buffer called *\*Index-Entries\**. When you execute one of these commands, this buffer appears in the top half of the screen, and the current buffer in the bottom. Clicking  $\leftarrow$  on an entry moves you to the record that generated that entry.

### Collect Index Entries of Buffer

Collects index entries for the documentation records in a buffer. The index entries are collected in the editor buffer *\*Index-Entries\**.

In addition, this command sets up split screen mode with *\*Index-n\** in the top window, and the previous buffer in the bottom window.

### Collect Index Entries of Tag Table

Collects index entries for the documentation records for the files that are part of the specified tag table. The index entries are collected in the editor buffer *\*Index-Entries\**.

In addition, this command sets up split screen mode with *\*Index-n\** in the top window, and the previous buffer in the bottom window.

### Collect Index Entries of Topic

Collects index entries for the documentation topic you specify. The index entries for the topic and its expanded records are collected in the editor buffer *\*Index-Entries\**.

In addition, this command sets up split screen mode with *\*Index-n\** in the top window, and the previous buffer in the bottom window.

## 19. Customizing Your Symbolics Concordia Documents

You can tailor Symbolics Concordia documents to meet immediate document viewing needs. Symbolics Concordia provides a set of tools that allow writers to create custom documents. Writers can define document formatting options that can be set in the document itself by the writer, or can be set later by the reader when the document is viewed online (or when the document is formatted to make a hardcopy book).

By defining appropriate document formatting options, writers can create intelligent documents that can deliver information based on realtime conditions such as the expertise of the reader (for a training document), or the status of the equipment (for a repair manual).

For example, writers could create formatting options for training manuals based on reader expertise. An expert would see only usage procedures; a novice would see conceptual and detailed descriptive information (as well as usage procedures).

Writers could also create formatting options based on equipment status. A technician could choose to see more detailed information when troubleshooting malfunctioning equipment, than when equipment is functioning properly.

Other formatting options might be based on job function, the current date, a version number, an equipment platform, a language, or a model number. Writers have complete flexibility with respect to creating document formatting options.

Custom documents include intelligent links that are expanded based on options that the viewer selects when the document is displayed or formatted for viewing. (Viewing options can also be set in the document itself.)

You define document formatting options by creating Case markup in your document (using the `=M` "Case Command" ).

Case environments contain a Case selector, and one or more named Case clauses. The Case selector names a variable, which establishes a particular viewing option for its associated Case clauses. A Case clause encloses documentation that is formatted for viewing when the value of its Case selector (variable) matches the name of the Case clause. For example, this Case environment defines the Expertise Case selector with two defined options, Novice and Expert.

```

|Case
  Selector: EXPERTISE
  |Novice
    ➤ Include link: Overview of Washing Your Cat (section)
    ➤ Include link: How to Wash Your Cat (section)
  |Novice
  |Expert
    ➤ Include link: How to Wash Your Cat (section)
  |Expert
    Click here to add a new clause
|Case

```

In this example, the String command sets the Expertise Case selector to the value Novice. When the document is viewed online (or when pages are formatted), any text within Novice Case clauses in the document is displayed.

```

|String
  |Expertise
    Novice
  |Expertise
    Click here to add a new string
|String

```

If you set the Expertise Case selector to any other value (Expert, for instance), different documentation is displayed.

Note that there are several ways to express formatting options for a document. The following markup would display the same documentation as above, but it expresses slightly different formatting options.

```

|Case
  Selector: EXPERTISE
  |Novice
    ➔ Include link: Overview of Washing Your Cat (section)
  |Novice
    Click here to add a new clause
|Case
  ➔ Include link: How to Wash Your Cat (section)

```

There are many ways to set Case selector variables. You can set them in the document, or you can set them when you view/format the document.

## 19.1. Setting Document Formatting Options in Symbolics Concordia

Set Symbolics Concordia document formatting options by assigning values to its Case selector variables.

Formatting options apply to all of the following document viewing and formatting methods:

- "Format Pages Page Previewer Command "
- "Show Documentation Command" (in the Page Previewer, Document Examiner, or the Symbolics Concordia editor)
- `s-P` (in Symbolics Concordia)

The formatting behavior of an option when the document is viewed/formatted is determined by how its Case Selector variable values are defined and set.



There are several ways to set Case selector values. You can set them in the document itself, or you can set them using Page Previewer, Lisp Listener, or Document Examiner commands.

Case Selector variable types are defined as follows:

- **Static Variables**

These variables are evaluated first and keep their values across formatting runs. You can define (and set) these variables using the "Set Sage Variable Command".

You can define static variable values in your document using the "PermanentString Command". (You can assign a value to a string with the "Value Command".)

Note that, when you use static variables to set formatting options, the document may format differently for Hardcopy Pages than it does for online viewing (Show Documentation or  $\varepsilon$ -P). When you view the document online, all PermanentStrings are evaluated first. When you use the "Format Pages Page Previewer Command" to make page images, PermanentStrings are evaluated as they occur in the document.

- **Ambient Values**

These Case Selector values are assigned only during a particular formatting (or online viewing) run.

You can define ambient variables using the "String Command" in a Symbolics Concordia record. (You can assign a value to a string with the "Value Command".)

Note that, when you use ambient values to set formatting options, the document may format differently for Hardcopy Pages than it does for online viewing (Show Documentation, or  $\varepsilon$ -P). When you view the document online, all Strings are evaluated first. When you use the "Format Pages Page Previewer Command" to make page images, Strings are evaluated as they occur in the document.

- **System Variables**

A predefined set of variables that writers can use as Case selectors. See the section "Sage System Variables", page 193.

These variables do not need to be set, they are guaranteed to have a value that is determined by the system.

- **Batch Formatting Options**

You can temporarily set formatting options for the document using the `:Query` option of the "Format Pages Page Previewer Command ". You can choose to be prompted for All, None, or only Unassigned Case selectors in the document.

This is a good way to hardcopy different versions of a document.

You can use this option to temporarily override any default Case selector values that are set in the document (or elsewhere). Note that these selectors remain assigned only for the duration of a single formatting run. This example formats pages for the document "Symbolics Concordia 3.1 Release Notes", querying for any Unassigned Case selectors. At the prompt, the Case selector variable `DOC TITLE` is set to the value "symbolics concordia".

```
Page Previewer command: Format Pages "Symbolics Concordia
3.1 Release Notes" :Query Unassigned
```

```
Enter value for Case selector DOC TITLE: symbolics concordia
```

The formatting options for this document are designed to produce an abbreviated version of the Release Notes for the Symbolics Concordia document, and an expanded version, otherwise.

Case selectors are evaluated in the above order of precedence (that is, Static Variables are evaluated first, and Batch Formatting Options are evaluated last).

This table summarizes the methods for setting Case selector variables. They are listed in precedence order (that is, variables set with `PermanentString` command are evaluated first, and variables set with `Format Pages :Query` are evaluated last).

<b>Concordia</b>	<b>Document Examiner</b>	<b>Page Previewer</b>	<b>Lisp</b>
PermanentString String	Set Sage Variable	Set Sage Variable	Set Sage Variable
			sage::register-book :sage-variables
		Format Pages :Query	

Because the exact order of evaluation of formatting options within a document is not guaranteed, we recommend that you set Case selectors in the top-level script record of a document.

### 19.1.1. Using Sage Static Variables to Set Document Formatting Options

You can use Sage static variables as Case Selectors to set Symbolics Concordia document formatting options. Sage static variables are evaluated first when the document is formatted/viewed.

There are many ways to set sage static variables:

**Concordia Editor**

"PermanentString Command"

<b>Document Examiner</b>	"Set Sage Variable Command"
<b>Page Previewer</b>	"Set Sage Variable Command"
<b>Lisp Listener</b>	"Set Sage Variable Command"
	<b>sage::register-book</b>
	<b>sage::set-static-value</b>

### 19.1.2. Using Sage System Variables to Set Document Formatting Options

Symbolics Concordia provides a list of predefined Sage system variables you can use to set document formatting options. You do not have to set these variables. The value is set by the system.

#### Sage System Variables

<b>Variable</b>	<b>Value</b>
Date	February 2018
Day	8
FileDate	8 February 118 at 16:23
Month	February
RootFileDate	8 February 118 at 16:23
Time	16:23
Timestamp	8 February 118 at 16:23
Weekday	Thursday
Year	2018

You can also create new Sage system variables using **sage::define-system-variable**.

```
(sage::define-system-variable name accessor)
```

*Name* is a string that identifies the sage system variable, and *accessor* is a function of zero arguments which returns a string.

For example,

```
(define-system-variable "machine type" #'machine-type)
(sage::define-system-variable "user id" #'(lambda () neti:user-id))
```

## 19.2. Concordia Example: A Book with Multiple Versions

This example shows how you can use formatting options to make multiple platform-specific versions of an installation guide.

In this example, the top-level script record defining the book uses the String Command to set the Platform Case Selector variable to "PC". Note that you can also

set formatting option defaults for the document using

```
(sage::register-book "Book with Multiple Versions" :document-type
'sage::3symanual :sage-variables '((Platform "PC")))
```

When the document is formatted/viewed, only PC Case clauses in the document are formatted.

Note that you can also use Case commands in the Display-Name field of the record to add platform-specific information to the displayed topic title.

### Markup

These screen images show the Symbolics Concordia markup that defines the formatting options for this document:

Section "Installing Your System on a PC"

Contents

This is the top level script record that defines the book. There are other script records that share this Include link. The Case selector variable Platform is set appropriately in each document script record.

String

Platform  
PC

Platform  
*Click here to add a new string*

String

► Include link: Installation Procedure (section)

Contents

---

Section "Installation Procedure"

Contents

This topic is formatted based on the value of the Platform Case selector variable. Text outside of the scope of the Case environment is platform independant.

Case

Selector: PLATFORM

Pc  
This contains PC-specific instructions.

Pc

Ux400s  
This contains UX400S-specific instructions.

Ux400s  
*Click here to add a new clause*

Case

Contents

### Display

This shows what is displayed when you `s-P` on the top level record:

**Installing Your System on a PC**

This is the top level script record that defines the book. There are other script records that share this Include link. The Case selector variable Platform is set appropriately in each document script record.

**Installation Procedure**

This topic is formatted based on the value of the Platform Case selector variable. Text outside of the scope of the Case environment is platform independent.

This contains PC-specific instructions.



## 20. Displaying, Reviewing, and Publishing Documentation

### 20.1. Displaying and Reviewing Documentation Online

Reviewing the contents of each record in the buffer is a necessary, but not sufficient, check of your work. By displaying a record or series of records, you can review the effect of markup diagrams and get a sense of how well the linked records fit together.

By means of the Symbolics Concordia editor or Document Examiner, you can display documentation on the screen that approximately matches its corresponding appearance in a printed book. In both contexts, the formatter processes the documentation: all links show the correct view, text is complete and exact, text markup is converted according to the instructions of the formatting environment; formatting commands are executed; and illustrations from the Graphic editor are shown scaled and in place. However, such niceties of printed books as page numbers, table of contents, and indexes are missing (for more information, see the section "Using the Page Previewer", page 201 ).

Proofreading in this manner is preferable to reading individual records in the buffer, because it mimics the way a real end user will read your documentation.

#### 20.1.1. Reviewing Work in the Symbolics Concordia Editor

Use `␣-P` in the editor to display a record. When you find an error, you can either abort the display, fix the error in the original record, and then redisplay the record, or you can note all errors and fix them in a batch, redisplaying the record only when you have completed all corrections. The point here is there is no way to fix an error and then return to the original display, resuming at the point you left off. See the section "Preview Records".

#### 20.1.2. Reviewing Work in Document Examiner

Reviewing work from Document Examiner provides an easy way to move between the original record in the editor buffer and the displayed record in Document Examiner.

Press `SELECT D` to go to Document Examiner. Use the Show Documentation command for the topic you want to review. Document Examiner uses the latest edited version, if it is available. Otherwise, the published version (the one that lives in the database) is read in.

You will have to switch from Document Examiner to the Symbolics Concordia editor to make your corrections, locating records with `␣-`. After making all corrections, you probably want to go back to Document Examiner and redisplay the edited records. First click Middle on [Viewer] in the command menu pane to clear the viewer, then use [Show Documentation] to request the topic again.

## Comparing Different Versions in Document Examiner

After reworking a record over a period of time, the record can change radically from its original, published version. Occasionally, you need to look at the published record to see what information you've added and deleted.

The mode-setting feature of Document Examiner makes it relatively easy to compare the "before" and "after" versions of your work. Modes are explained in "Lookup Modes and Actions in Document Examiner".

Follow this procedure to compare two versions of a record.

1. Select Document Examiner. Assuming that the most recent version of the record is read into your world, display the latest version of the record in the current viewer by clicking on the Show Documentation command and typing the name of the topic.
2. Select another viewer and invoke Set Lookup Mode to set the mode of the current viewer to reader. (As a rule, it's not a bad idea to create one reader-mode viewer for just such comparison purposes.) Again click on the Show Documentation command and type the name of the topic.

### 20.1.3. Proofreading Your Work

When you proofread your work, make sure of the following:

- That text within a record is modular.
- That text reads smoothly, in logical order and with good transitions.
- That text contains no grammatical or typographical errors.
- That all records that should display do display, as described in "Examining the Organization of Your Document".
- That crossreferences are correct.
- That formatting environments and commands give you the results you want.

### 20.1.4. Displaying Documentation

Sometimes you need only to display documentation — not for proofreading purposes, but to see that the contents are generally what you expected. For example, you want to make sure that you are adding a crossreference to the correct record. Here are some quick ways to display the records that are not current in your buffer:

- Click Left on a record name in the Collected Record Names pane to display the formatted documentation in a typeout window;
- Click Right on a record name in the Collected Record Names pane; a menu pops up, one of whose options is Show Documentation of that record.



- Use `m-X` Show Documentation. Type a record name or click Left on a record name in the buffer.

### 20.1.5. Printing Draft Copies of a Document

Writers often use the Symbolics Concordia editor to produce *informal* review copies; that is, a hardcopy of the text but no index, table of contents, page headings, and so on. To print a topic (the contents of a record and all its links), position the cursor on the record and use `c-U s-P`. You are prompted for a printer.

## 20.2. Publishing Your Documentation

Eventually, you'll want to make your formatted document available to other Genera users. Unless you explicitly publish your documentation, that is, add it (its constituent files) to a documentation system, the records in your files are available only to you and anyone else who knows the pathnames of your files.

Documentation is added a documentation system either by recompiling the documentation system or by *patching* records (incrementally adding to/fixing the database). This published documentation is what most Document Examiner users see when they look up any part of your document.

Documentation patches are made in the same way that patches are made to the rest of the Genera software. For further information on documentation systems and patching, see:

- "Guide to Documentation System Maintenance"
- "Patching a Documentation System"

In addition to the standard patching commands, you can use these Symbolics Concordia commands to patch documentation systems:

- "Add Patch Changed Records"
- "Add Patch Changed Records of Buffer"

#### Add Patch Changed Records

Adds all changed, unpatched records from all buffers to the current patch. If no patch is open, this command starts a patch. You are asked to choose one of these options:

- Y Patch this record
- N Don't patch this record
- P Proceed and patch all records

### Add Patch Changed Records of Buffer

Adds all changed, unpatched records in the current buffer to the current patch. If no patch is open, this command starts a patch. You are asked to choose one of these options:

- Y Patch this record
- N Don't patch this record
- P Proceed and patch all records

If you distribute your documentation system online, see the section "Guide to Documentation System Maintenance", page 431 for information on setting up and distributing a documentation system.

## 21. Using the Page Previewer

### 21.1. Overview of the Page Previewer

The Page Previewer is a page layout tool that prepares documents for formal output. Documents produced with the Page Previewer have a table of contents, index, section numbering, and the usual characteristics of a book. Therefore, you will want to use this facility to produce hardcopy of the final version of your document.

The Page Previewer shows you the layout of your document, providing a facsimile of how your document (or any part of your document) will appear when printed. The spacing and fonts you see in the Page Previewer differ from those that appear in the printed document. The placement of words and illustrations, however, is exactly the same.

The screen preview shows you the *shape* of the page rather than its actual contents, providing you with the opportunity to identify and fix design problems — bad line or page breaks, for example — *prior* to printing. The Page Previewer is not intended for proofreading, but rather as an aid to locating formatting and pagebreak problems.

Click on the Page Previewer icon, the middle icon in the Icon pane or use `⌘-SELECT P` to get to the Page Previewer. Here's a brief explanation of how to use the Page Previewer.

### 21.2. Producing a Document as a Book

When you do `Format Pages` in the Page Previewer, Symbolics Concordia formats your document for paper output. Unless you have declared your document to be a *book*, the top-level record in your document is treated as a chapter and the records it links to are treated as sections in that chapter. If you *register* your document as a book, by evaluating a `sage::register-book` form, your top-level record is treated as a book and the records it links to are treated as the *majorparts* of that book. A simple `sage::register-book` form might look like this:

```
(sage::register-book "Scheduler Documentation")
```

The string, "Scheduler Documentation" in this example, should be exactly the same as the topic name in your top-level record. This is the bare minimum to get "Scheduler Documentation" treated as a book. Usually you want to specify a document type and indicate how your document is structured. Specifying a document type indicates which book design will be used to format your document.

```
(sage::register-book "Scheduler Documentation"
 :document-type 'sage::3symanual)
```

`3symanual` is the default document type for Symbolics documentation, and initially the default for Symbolics Concordia. The default at your site can be set to your

site's book design by setting the variable **sage:\*default-document-type\***. (See the section "Defining a Book Design System for a Site", page 410.)

The default for the structure of a document is to have several *majorparts* which are then divided into chapters (for example, *Symbolics Common Lisp Language Concepts*, book 7 in the Symbolics documentation set). If your book is only one structural unit, and you want the records linked to by your top-level record to be chapters, your **sage::register-book** form should look like this:

```
(sage::register-book "Scheduler Documentation"
  :document-type 'sage::3symanual
  :highest-structural-level 'sage::chapter)
```

### 21.3. Formatting Pages

To format and display a record, use Format Pages and supply the name of a record. To format the entire document, supply the name of the top-level record.

**Note:** Books with forward references must be reformatted until these references are resolved. Generally, you need to format a document with forward references twice, but depending on the complexity of the structure of your documentation set, you may need to reformat a document several times. The Page Previewer informs you of any unresolved references (also known as "improperly referenced tags"). (You can use the Show Improperly Referenced Tags command to display a list of currently unresolved references.)

You can use the Save Tags command to save the page references across sessions to avoid having to rerun the formatter each time you format your document. Use the Restore Tags command to reload the page references for a topic when you want to format it the next time.

The previewer formats the table of contents, the list of figures, the text, and the index, in that order.

The first page of the formatted text, page 1, is displayed on the screen, just as it would appear in the printed book.

Odd numbered pages appear on the right, even numbered pages on the left. Beneath the formatted page, the previewer indicates the current page and the total page count. For example, 1/82 means you're looking at page 1 out of a total of 82. (The page count includes the text, the table of contents, and the index.)

### 21.4. Resolving Crossreferences

A Tag is an internal token that the formatter uses to store page numbers to make crossreferences. As it formats each record (or Tag markup in a Figure or Table) it creates an internal tag consisting of the record name or the value of the Tag markup and the page number. When it sees a crossreference (or Ref markup), it searches in the table of tags for the tag that matches it and puts the value of the

tag in the output. If the crossreference to something comes first, it just hangs on to the reference. In due course it finds the record (or figure) and creates the tag. However, it cannot put the value of the tag in the output on this pass because it proceeds linearly through the document, and the location of the reference is behind it. When the format run finishes, the formatter announces the total number of these "improperly referenced tags" it is still holding on to.

If you look through the formatted pages in the Page Previewer, you find crossreferences that look like this when a tag has been "improperly referenced":

See the section SYMBOLICS-CONCORDIA-WRITER'S-GUIDE-SECTION,  
page SYMBOLICS-CONCORDIA-WRITER'S-GUIDE-SECTION.

You should run the formatter another time before printing out the document, to allow it to resolve these references.

Note: the tags that stand-in for page numbers are long, and usually get replaced with from one to three digits. Sometimes the shortening of lines on a page as these tags are replaced with numbers changes the pagination, which can change some references that have already been resolved. This means that it is possible to run the formatter again and get a different set of improperly referenced tags. With a large document (several hundred pages), anywhere from three to six runs may be necessary to propagate the page changes through the book and resolve them all. We recommend that you continue running the formatter until there are no more messages about improperly referenced tags, or you get the same list of improperly referenced tags twice.

If there are more than half a dozen improperly referenced tags, the list is not displayed automatically. Use the command Show Improperly Referenced Tags to see them all.

When you get the same list of unresolved tags on two successive formatter runs, you probably have crossreferences to records that are included by Contents links. Tags for records are generated by the header (or topic name), and records linked in by contents links do not have headers (that being the purpose of the contents link), and hence do not generate tags. The thing to do is to make the crossreference to the record that contains the contents link. Using "Symbolics Concordia Writer's Guide" as an example,

1. Do Show Links To Record. You see something like this:

3 links to "Symbolics Concordia Writer's Guide" from other records:

As Crossreference:

Symbolics Concordia Workbook: Creating Graphics

Symbolics Concordia Workbook: Viewing Your Document in the Page Previewer

As Contents:

Symbolics Concordia

2. Go to each of the records that have a crossreference to "Symbolics Concordia Writer's Guide" and change the link to be to "Symbolics Concordia".

To change a link you do not have to delete it and make another one. Click `C-M-Right` on the crossreference links, as you would if you were getting rid of the initial capital letter. One of the things in the menu is the name of the record the link is to. Click `Left` on that and type in the new record name.

Do this for each of the improperly referenced tags, `Format Pages` again, and `Hardcopy`.

When you have resolved all the references for a document, it is a good idea to use the `Save Tags` command to save the tags and associated page numbers in a file. Then, when you want to run the document off again, you can use `Restore Tags` to read in the old tag information and use that as a starting place. This usually saves you several formatter runs.

Occasionally the Formatter reports that it has two or three improperly referenced tags when, in fact, everything is resolved. This usually happens when you have formatted something several times and then have removed some crossreferences. The Formatter still has a tag for the topic, but it can no longer find the reference to it. So it continues to report it as improperly referenced.

## 21.5. Moving to Another Page

Move from page to page with `Next Page` and `Previous Page`. If you prefer to use the mouse, click on the black, rectangular *page holders* on the corners of the pages, shown in figure 43. Clicking on the left and right page holders will move you forward and back one page, respectively.

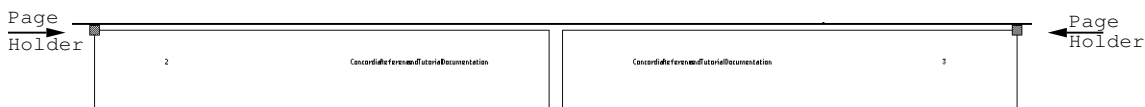


Figure 43. Click on the page holders to move from page to page.

To go directly to a specific page, use the `Set Page` command and supply a page number. Note that you must offset the page number by the number of pages in the table of contents and list of figures.

Entries in the index, table of contents, and list of figures are mouse-sensitive. To display a particular section, go to the table of contents. Click on the entry of your choice to have its associated record displayed. Click on an entry in the index or list of figures to move to the record containing that index entry or figure, respectively.

**Note:** If you prefer to see one page at a time, rather than facing pages (the default), use the `Change Layout` command.

## 21.6. Fixing Formatting Errors

The individual records composing the formatted pages (that is, the formatted top-level record and all its linked records) are mouse-sensitive. When you spot a formatting problem, either click `m-Left` on the erring record or type `Edit Record` and supply a record name (type it or click `Left` on the name). Symbolics Concordia returns to the editor, with the cursor positioned at the beginning of the record. Fix the problem, then return to the Page Previewer and rerun the formatting job from the beginning.

If you're not satisfied with the way the Page Previewer handled widows and orphans, use `Set Widow Action` and then reformat the record. For information on `Set Widow Action`, see the section "Page Previewer Commands", page 206.

Although the Page Previewer is best suited to handling layout and formatting problems, you might also take the opportunity before final printing to check out the structure of your document and/or suspicious-looking text.

You can take a closer look at sections of text in the Page Previewer by using the `Show Documentation` command. Click `Left` on the record or type the command at the prompt and supply a record name. If you decide to return to the editor to make any changes to the record, remember to rerun the formatting job.

You can examine how records are linked with `Show Overview`. The command shows you how a particular record fits into the document structure. Note that `Show Overview` works on any installed record. You do not need to run `Format Pages` before using `Show Overview`.

## 21.7. Displaying Previously Formatted Pages

`Format Pages` displays the last record you have formatted. The Page Previewer stores all records formatted during the current boot session. To see a previously formatted record, use `Show Formatted Pages` and type the name of a formatted record. If you do not remember the names of the record, press the `HELP` key to see a complete mouse-sensitive list of formatted records.

**Note:** This command does *not* reinvoke the formatter; it merely displays the results of a previous formatting run.

## 21.8. Printing Formatted Pages

The Page Previewer provides several ways to get paper copies of your work.

- Use `Format Pages` with the `:printer` option to format a record and immediately send the results to a printer. Obviously, this method does not provide the opportunity to examine your work first before printing it.

- After running Format Pages, use Hardcopy Pages to send one or more pages to the printer of your choice.
- For a previously formatted record, use Show Formatted Pages with the **:printer** option to format a record and immediately send the results to a printer. Use Show Documentation, with a **:destination** option, to hardcopy a record and all its linked records. This method prints the text and illustrations, but no page headings or footings or page numbers.

## 21.9. Page Previewer Commands

You type commands at the prompt Page Previewer Command:. Press HELP to see a complete list of comands. For information on how to enter a command, see the section "Communicating with Genera" in *Genera User's Guide*.

**Change Layout** Changes the layout from two-page (the default) to one-page, and back again.

**Edit Record** Locates the specified record, taking you from the Page Previewer into the editor.

**Format Pages** Formats and displays the specified record. When it prompts for a topic, supply the name of the record you want to format.

Note that Format Pages evaluates String and PermanentString commands as they are encountered during the formatting process (that is, they are not evaluated first). Use care when you use these commands to set conditional document formatting options. See the section "Customizing Your Symbolics Concordia Documents", page 189.

Format Pages takes these keywords:

**:index** {Yes or No} default Yes

Produces an index for the formatted topic. Use :do index no when you do not want an index.

**:toc** {Yes or No} default Yes

Produces a table of contents for the formatted topic. Use :do toc no when you do not want a table of contents.

**:printer** *printer-name*

Sends the output to the printer of your choice.

**:query** {All, None, or Unassigned}

Controls prompting for document formatting options when the document is formatted.

All prompts for all Case selector values (including those that are already set in the document). Note



that the Page Previewer only prompts once for the value of a Case selector (that is, the first time it is encountered in the document). For more information, see "Case Command".

None does not prompt for new Case selector values.

Unassigned prompts only for "unassigned" Case Selectors. Note that the Page Previewer only prompts once for the value of a Case selector (that is, the first time it is encountered in the document).

At the end of each formatting run this command displays any Improperly Referenced Tags (unresolved topic cross references), and any Unassigned Case selectors in a typeout window.

The Unassigned Case selectors display looks like this:

```
There were 2 unassigned Case selectors:
Unassigned Case selectors for "Mantis Quarterly Report" Section:
OVERHEAD STATUS in "Expenses" Section
DETAIL in "Building Report" Section (no Unassigned clause provided
```

Each entry in the list shows the name of the unassigned Case selector, the record in which the Case is located, and whether or not there was an Unassigned clause in the Case command. This is useful since if the Case selector is unassigned and there is no Unassigned clause, nothing in the Case environment is formatted.

Note that the record names in the list are mouse sensitive. You can click  $n$ -Left to edit them.

When there are many Unassigned Case selectors, the following message is displayed (You can click to see the detailed display):

```
There were 23 unassigned Case selectors. Click here to see them.
```

### Hardcopy Pages

Sends one or more formatted pages to the printer. It prompts for a formatted record group. Supply the name of a record that has been run through Format Pages. It also prompts for beginning and ending pages (the defaults are 1 and All, respectively) and a printer name. It takes two keywords:

	<b>:background</b> {Yes or No} default Yes	Runs the hardcopy process in the background, so you can switch back to the editor and continue work.
	<b>:to file</b> <i>pathname</i>	Stores the formatted pages as a postscript file for printing later.
Next Page		Displays the next page of a formatted topic.
Previous Page		Displays the previous page of a formatted topic.
Restore Tags		Takes the pathname of a tag file and reloads the tags into the world.
Save Tags		Takes a pathname and saves the tags from the formatting of a topic in that file.
Set Page		Displays the formatted page of your choice; prompts for a page number.
Set Widow Action		Controls the handling of widows and orphans. The Page Previewer takes one of the following actions:
	Ignore	Breaks the page anyway.
	Warn	Breaks the page anyway and also prints a warning.
	Force	Moves the orphan into the margin of the next page or moves the widow into the margin of the previous page.
	Forcewarn	Acts like Force and also prints a warning.
Show Documentation		Prompts for a record and displays its installed documentation in a typeout window. Also available by clicking Left on a record in the Previewer. Use the <b>:destination</b> keyword option to send output to the printer of your choice. Note that Show Documentation prints text and illustrations but does not print section numbers, page headings, and so on.
Show Formatted Pages		Displays a previously formatted record. When it prompts for a formatted record group, supply the name of a record that has been run through Format Pages. Use the <b>:printer</b> keyword option to send the output to the printer of your choice. Press HELP to see a mouse-sensitive list of all the records you have formatted during the current boot session.
Show Overview		Prompts for a record and displays its overview, showing how it fits into the overall document structure.

### Show Improperly Referenced Tags

Displays the list of unresolved tags..

You can use these commands to set document formatting options:

- **Set Sage Variable Command**  
Sets the named sage variable to the specified string value. Use sage variables as Case selectors to control document formatting options.
- **Clear Sage Variable Command**  
Clears the named sage variable.
- **List Sage Variables Command**  
Displays a list of the sage variables that are currently set.

Additional commands available from the Page Previewer are:

- **Delete Printer Request Command**
- **Restart Printer Request Command**  
Restarts a print request that has not yet finished.
- **Show Printer Status Command**  
Displays the print queue for the specified printer or printers.



## 22. Workstyle Issues in Writing with Symbolics Concordia

### 22.1. Introduction to Modular Writing

When preparing *printed* books, you probably make certain assumptions about how readers use your documents, which in turn affects how you write books. For example, you might take for granted that readers read from front to back — they skim chapters, then sections, and finally subsections; they probably skim early pages before later pages. Books often contain phrases like "As shown in the example above", where the example lives in another section altogether. As a result of assuming a linear reading order, you might also assume that readers always have the right context. Even if they read a given subsection in isolation, they can easily skim back over previous pages to get the necessary background material.

When preparing documentation for online lookup (as well as for the printed page), you must acquire a new set of assumptions. The most important is that readers look up documentation in Document Examiner out of context. Each record must make sense when read on its own, without reference to material covered in other sections. In short, modularity is a fundamental requirement of Symbolics Concordia.

### 22.2. An Example of Writing for Modularity

The introductory paragraph of a section called Logical Pathnames assumes familiarity with another kind of pathname, known as a physical pathname, but does not explicitly refer to it.

#### Logical Pathnames

There is *another kind of pathname* that does not correspond to any particular file server. It is called a "logical" pathname, and a host called a "logical" host. Every logical pathname can be translated into a corresponding "physical" pathname; there is a mapping from logical hosts into physical hosts used to effect this translation. [*Emphasis added.*]

In a printed book the above paragraph may be more comprehensible because of surrounding text. For instance, if the immediately preceding section explains physical pathnames, readers can make a connection between sections. For the purpose of online lookup, however, the paragraph should be rewritten to refer to the other kind of pathname by name.

The next paragraph shows an improved version of this example. An introductory phrase provides a context for readers who look up this section in isolation from the discussion of physical pathnames. Readers can then search for information relating to physical pathnames in Document Examiner.

#### Logical Pathnames

*In addition to physical pathnames, there is another kind of pathname that does not correspond to any particular file server. It is called a "logical" pathname, and a host called a "logical" host. Every logical pathname can be translated into a corresponding "physical" pathname; there is a mapping from logical hosts into physical hosts used to effect this translation. [Emphasis added.]*

### 22.3. Using Crossreferences in Modular Writing

As a further refinement, you can also place a crossreference to a record that describes physical pathnames in the Logical Pathnames record.

You can use crossreferences as a tool to enhance the power of individual records. Crossreference topics are mouse-sensitive and can be used as stepping stones by readers to proceed from one piece of information to the next.

In our example, we can add a standard crossreference to the record Physical Pathnames.

#### **Logical Pathnames**

In addition to physical pathnames, there is another kind of pathname that does not correspond to any particular file server. See the section "Physical Pathnames". ...

The effect is a mouse-sensitive phrase that appears within the regular text. The reader can see documentation for that topic by clicking on the highlighted phrase. The only restriction in creating a crossreference link is that the phrase must correspond exactly to the name of a record.

In addition to a standard form of crossreference, you can use the topic view of a crossreference in text.

In the next example, Physical Pathnames is the name of a record. The writer has embedded the name, using the topic view of a crossreference, in the opening sentence. Readers can click on "Physical Pathnames" to see the documentation for that record.

#### **Logical Pathnames**

In addition to "Physical Pathnames", there is another kind of pathname that does not correspond to any particular file server. ...

The availability of crossreferences ensures that online readers can be given handy access to important background or auxiliary information.

For further information on how to create crossreferences, see the section "Link Commands", page 105.

## 22.4. Problems and Solutions for Modular Writing

### Indefinite Crossreferences

Indefinite crossreferences assume a linear reading style and should be avoided. Some examples of indefinite crossreferences:

"As stated earlier in the chapter, programs can handle errors by..."

"See below"

"See above"

"The introduction to this chapter explains so-and-so..."

### Multi-purpose Records

Another common assumption is that divisions of a printed book are organized hierarchically into chapters, sections, subsections, and so on, and that book divisions are usually numbered to indicate each division's place in the hierarchy. Document Examiner, however, ignores the visual hierarchical distinctions; for example, topics are not displayed with numbers. The reason is that records are reusable. A section in one printed book might appear as a subsection in another.

Avoid phrases that a "hardwire" a section's place in a book, because any record can be used in more than one document.

Bad Example: "The first section of this chapter explains so-and-so". "The introduction to this chapter *explained* how the system handles errors. This section explains the system's debugging facilities."

The above examples can be made more specific, and less context-dependent, by using a crossreference. Note that the rewrite avoids the use of the past tense "explained", which assumes the reader has already seen that section.

Good Example: "This section explains the system's debugging facilities. For general information on how the system handles errors: see the section 'Errors and Debugging'."

### Some General Rules

To help ensure modularity:

- Use running examples carefully.
- Ask yourself whether the text of your record relies on concepts introduced in other records. If so, consider inserting a crossreference.
- Ask yourself whether you have too much information in any one record. Can you break out some of the text into other records?
- If you are considering using a Heading or Subheading command, ask yourself if the text to be placed under the heading or subheading really should be in a record of its own.





## 23. Symbolics Concordia Customizations for Your Init File

### 23.1. Customizing the Editor Pane

The Editor Commands pane can be customized in several ways: the entries under one or more categories can be hidden from view (collapsed) or the categories themselves can be reordered so that the groups most useful to you are at the beginning of the menu. For instructions, see the section "Customizing the Symbolics Concordia Window", page 61. To have the menu customized automatically when you log in, click Left on **Editor Commands** at the top of the pane. Then select Save Set in the menu that appears. Save Set pushes a form suitable for yanking onto the kill ring. You can then yank this form, (**sage::setup-editor-menu ...**), into your `lisp-init` file.

You can also put the cursor on a blank line near your other Zwei/Concordia initializations and execute the `C-M-X` Insert Concordia Menu Choices command.

### 23.2. Setting the Lookup Mode in the Document Examiner

To have the lookup mode set when you log in, set the variable **sage:\*record-lookup-mode\*** in your init file.

**sage:\*record-lookup-mode\*** *Variable*

Controls what version of the documentation is displayed when you look up topics. The possible values are:

**:normal** Uses the version of the record that exists in the Symbolics Concordia editor buffer. This is the default.

**:use-published-version** Uses the published record. If no published record is found, uses the edited record.

**:edit-newest-record** Uses the newest version of the record. Finds the most recent version of the sab file containing the record and reads it into the editor.

Example:

```
(setq sage:*record-lookup-mode* :edit-newest-record)
```

In Document Examiner you can set the lookup mode with Set Lookup Mode.

### 23.3. Setting the Buffer Locking Mode in Symbolics Concordia

To have the buffer locking mode set when you log in, set the variable **zwei:\*concordia-buffer-disposition\*** in your init file.

**zwei:\*concordia-buffer-disposition\*** *Variable*

Allows you to set the action you want for file locking in your init file. This eliminates the query about buffer disposition when you modify a .sab file.

The possible options are:

<b>:ask</b>	The default. Prompts for buffer disposition.
<b>:disconnect</b>	Disconnects the buffer.
<b>:lock</b>	Locks the file for exclusive writing.
<b>:error</b>	Modifications to Symbolics Concordia buffers are not allowed. (This is synonymous with making Concordia buffers read-only.)

To have .sab buffers automatically locked, put this form in your init file:

```
(zwei:set-zwei-variable 'zwei:*concordia-buffer-disposition*  
                        :lock :where :global)
```

This variable can also be set with the `m-x Set Variable` command.

## 24. Recovering From Errors in Symbolics Concordia

### 24.1. When Symbolics Concordia Misplaces the Cursor

Most errors that put you in the debugger occur because Symbolics Concordia has lost track of the cursor. This happens most commonly when you are killing records or removing or yanking markup. The movement of editor structures relocates editor nodes, and in the course of this relocation, the cursor sort of slips into the cracks.

The most common error messages from the debugger in these cases are:

- **Point not in interval being displayed.**
- **Window start-bp not in interval being displayed.**

If you use `ABORT` to exit the debugger, the editor attempts to redisplay the buffer. However, in order to do that it must first locate the cursor, since redisplay tries to put the cursor location in the center, vertically, of the screen. Since it cannot find the cursor, you end up in the debugger again.

Try these steps to recover from the error:

1. First, try `RESUME`. This tries to relocate the cursor and continue. It might work.
2. If `RESUME` does not work, select the resume option `Skip Redisplay`. This exits from the debugger but does not try to redisplay the buffer.
3. Press `m-<` to jump to the top of the buffer. Almost all cursor motion commands are relative, and must first locate the cursor in order to know where to move it. However, there is one location in the buffer that is fixed, the top of the buffer. The editor does not need to calculate the current cursor location in order to jump there, so that is almost guaranteed to succeed. Once you are at the top of the buffer, when the redisplay happens the cursor is in a known location.

If you are still going into the debugger after skipping redisplay and jumping to the top of the buffer, try one of the following:

- Try saving the buffer with `c-X c-S`, `m-X Save File Buffers`, or from the editor menu.
- If after the buffer is saved you still go into the debugger, you probably need to kill the buffer and read the file in again.

- If you have not made any changes to the buffer, or have made only minor changes that you can easily redo, use `m-X` Revert Buffer to start off fresh.
- If the buffer is a newly created one that you have not invested much effort in yet, use `c-X c-K` to kill it and try again.

## 24.2. When Markup Damages the Undo History

Occasionally killing markup with `c-K` or `c-W`, or yanking it with `c-Y`, damages the Undo History. The Undo History is the record of all the changes (insertions and deletions) you have made to this buffer since you last wrote it to disk. Every now and then the editor chokes on a piece of markup or a link in this history. When this happens, you get this error message in the debugger:

**The array given to the ARRAY-LEADER instruction, "", does not have a leader.**

Pressing `ABORT` gets you out of the debugger. However, as soon as you attempt to insert something or delete something, you go right back into the debugger again. This is because any insertion or deletion gets pushed on the Undo History, which has been damaged.

To recover:

1. Press `ABORT` to get out of the debugger.
2. Save the buffer with `c-X c-S`, `m-X` Save File Buffers, or from the editor menu.

Saving the buffer clears the Undo History, repairing the damage.

## 25. Dictionary of Symbolics Concordia Editor Commands

This section describes the commands specific to the Symbolics Concordia editor. For information on other editing commands that you can use with Symbolics Concordia, see the section "Symbolics Concordia Reference Card", page 231 , and the "Reference Cards: Zmacs".

### Add Patch Changed Records

Adds all changed, unpatched records from all buffers to the current patch. If no patch is open, this command starts a patch. You are asked to choose one of these options:

- Y Patch this record
- N Don't patch this record
- P Proceed and patch all records

### Add Patch Changed Records of Buffer

Adds all changed, unpatched records in the current buffer to the current patch. If no patch is open, this command starts a patch. You are asked to choose one of these options:

- Y Patch this record
- N Don't patch this record
- P Proceed and patch all records

**Add Record Field** Adds a field to the current record, prompting for a field name. If the field already exists, it positions the cursor to the field in the record. The valid fields are Record-Name, Display-Name, Notes, Contents, Oneliner, and Keywords.

This command is available as Add Record Field under **Records** and as  $m-X$  Add Record Field.

**Break File Lock** Forcibly unlocks a file. This command is provided for emergency situations when a file cannot be unlocked by the conventional means of writing it out. The most common situation where this command is needed is if a machine crashes while it has a file locked. It is then necessary to break the lock so that another user can edit the file.

### Change Environment

Modifies the attributes of the nearest environment enclosing the cursor.

A menu pops up showing you the attributes of the environment with its current values. You can change any attribute, includ-

ing the environment name. If you change the name, the menu is updated, displaying the common attributes for the new environment, and markup for the new environment is displayed in the record. An environment that has been modified in other ways displays an ellipsis (...) in its markup.

This command is available as Change Environment under **Markup** in the menu or by clicking `C-M-Right` on the markup.

#### Clear Record Name Prefix

Clears the record name prefix prepended to newly created records. See also the "Set Record Name Prefix Command".

#### Clear Record Name Suffix

Clears the record name suffix appended to newly created records. See also the "Set Record Name Suffix Command".

#### Collect Record Name

Places a record name in the Collected Record Names pane. Collect the names of those records that you want to link to other records. When the "Create Link" command prompts you for a record name, you can click Left on a record name to select it from this pane.

#### Convert Flat Text To Record

Creates a new record from the marked region. The title of the record is the first non-blank text line in the region. The rest of the region is the contents of the new record. You are prompted for the record type.

The new record is placed immediately after the current one. The marked region in the current record is replaced with a link to the newly created record. It prompts for a view for the link.

#### Count Records in Buffer

Displays the number of records in the current Symbolics Concordia buffer.

#### Create Link

Creates a link to an installed record, at the cursor position. The cursor must be within the Contents field of a record. It prompts for a record name. You can click on one of the names in the Collected Record Names pane or type in a name.

It also prompts for a *view*, which determines which record fields get printed or displayed when the record is processed by the formatter. Each type of link produces its own view of a record.

Please read the background information provided about links. See the section "Link Types", page 86.

<i>Type</i>	<i>Description</i>
Include	View = Name and Contents fields.  Useful for incorporating text into and organizing the structure of a document. It creates visible structure (numbered/labelled sections) in a printed book.
Contents	View = Contents field.  Useful for incorporating text into a document. In this respect, it is similar to an include link. The main difference is that it blends text without displaying or printing the name of the record. Unlike an include link, it does <i>not</i> contribute to the visible structure of a printed book.
Crossreference	
	A crossreference has three <i>appearances</i> , Topic, Invisible, and See.
Topic	Name only. Useful for embedding a topic name in a sentence. You are responsible for spacing around the name.
Invisible	Does not display. Used in conjunction with the CrossReference environment to cause arbitrary pieces of text to be mouse sensitive. See the section "Invisible-style Cross-references", page 91.
See	Name and type, embedded in the sentence or sentence fragment, "See the <i>record-type record-name</i> ." You can specify Initial Cap for the word "See" and Final Period for the period at the end of the sentence by editing the link with <code>m-x Edit Link</code> or by clicking <code>c-m-Right</code> on the link.  When the record is printed on paper, the crossreference displays the location of the information: <ul style="list-style-type: none"> <li>• If the linked record and the linking record are included in the same book, the page number is printed.</li> <li>• If the linked record and the linking record are <i>not</i> included in the same book, the title of the book containing the record is printed.</li> </ul>

A crossreference link is useful for directing readers to additional information.

When the record is displayed, the crossreference is mouse-sensitive and looks as follows. Examples:

See the section "Putting Pictures in Text".

See the variable "**cp:prompt\***".

Note that Symbolics Concordia inserts the proper capitalization and punctuation. You, however, have responsibility for correct spacing.

**Precis** View = Name, Arglist (if the object is a Lisp object), and Oneliner fields.

Prints or displays the Lisp object name and argument list on one line and the oneliner summary on the next.

Useful for creating brief, dictionary-like lists or definition tables. Often used with environments to create formatted tables.

Clicking Left or Middle on Create Link in the menu provides a short-cut way to create an include or crossreference link, respectively. Clicking Right displays a menu of all link types.

This command is available as  $m-x$  Create Link or as Create Link under **Links** in the menu.

### Create Link and Record

Creates a record after the current record and leaves a link to it at the cursor location. It prompts for a record name, a record type, and a kind of link. (See "Link Types" for a description of link types.) If the cursor is not inside a record, a link is not created, and a warning to that effect is displayed.

If you mark a region and use this command, the region becomes the contents field of the new record.

This command is available as  $m-x$  Create Link and Record.

### Create Record

Adds a new record to the current buffer, prompting you for a name and a type (section, function, and so on).

When a region is marked, Create Record kills the region from the current record and yanks it into the new record. The new record appears immediately after the current record. The command creates a link in the original record to the new record, prompting you for a link type. (See the section "Link Types", page 86.)



- Create Record adds the name of the new record to the Collected Record Names pane.
- This command is available as Create under **Records** in the menu, or  $m-x$  Create Record.
- Edit Link** Pops up a menu to change the attributes of the link at the cursor location. You can change the view of the link, the appearance of crossreference links, or even the topic to which the link points.
- This command is available as  $m-x$  Edit Link and by clicking  $c-m$ -Right on the link to be edited.
- Find Link** Prompts for a documentation topic and for a link view (include, contents, crossreference, precis, or all views) and does a forward search for a link of that name and view. If it finds a matching link, it positions the cursor at the link.
- See also: "Reverse Find Link"
- Find Markup** Prompts for a markup environment (for example, Commentary, Figure, or Enumerate). It then searches for a markup environment of that type. If it finds one, it positions the cursor on the environment diagram line for the environment. Both the beginning and ending environment diagram lines are found by this command.
- Find Markup String** Prompts for a string and searches the structural markup and formatting markup in the buffer for any markup containing the string. If it finds the string, it positions the cursor on the beginning of the markup line. Repeating  $s-S$  and  $s-R$  finds additional instances of the string.
- $s-S$  maintains its own string history and offers you the last string used by Find Markup String. To repeat the previous search, you just accept this default, that is, press  $s-S$  followed by RETURN.
- This command is available as  $s-S$ .
- Find Orphan Records** Finds all records that either have no links to them or no contents regardless of whether they are linked. It searches all documentation systems you have loaded in your world.
- Find Orphan Records in Buffer** Find all orphan records in the buffer. An orphan record is one that has no links to it, or one that has no contents regardless of whether it has links.

### Find Orphan Records in Tag Table

Find all orphan records in the tag table. An orphan record is one that has no links to it, or one that has no contents regardless of whether it has links.

You can create a tag table in several ways:

#### Select All Buffers As Tag Table

Selects all buffers currently read in. Invoked by: `m-x Select All Buffers As Tag Table`.

#### Select Some Buffers As Tag Table

Selects buffers currently read in, querying about each one. Invoked by: `m-x Select Some Buffers As Tag Table`.

#### Select Some Files As Tag Table

Selects some files as a tag table. Read successive pathnames from the mini-buffer. Invoked by: `m-x Select Some Files As Tag Table`.

#### Select System As Tag Table

Selects all the files in a system as a tags table. Invoked by: `m-x Select System As Tag Table`.

If you have several tag tables defined, you can select the one to use:

#### Select Tag Table

Makes a tag table current for commands like Find Orphan Records Invoked by: `m-x Select Tag Table`.

### Graph Links from Record

Shows graphically all the include links *from* the record you specify *to* other records. It prompts for a record name; the default is the last record for which you displayed an overview in the editor or Document Examiner.

In effect, the display shows a table of contents for the specified record: which records are included in a particular record. This command lets you verify the structure of your document, confirming that all records are linked that should be linked.

This command is available as `m-x Graph Links from Record` and as Graph Links from Record under **Links** in the menu.

### Insert Record

Inserts a record, removed from a buffer with Remove Record From Buffer, in the current buffer.

### Kill Record

Kills the specified record, removing it from the Record Registry. You are informed of any links to the record. These links must be removed before the record can be killed. After the

record is killed, the record trailer and header remain with the label that this is an undocumented topic. You must patch the undocumented topic to remove it from the world. For further information on patching, see the section "Patching a Documentation System", page 442. This command is available as **Kill** under **Records** in the menu or as `m-x Kill Record`.

#### List Changed Records

Displays the names of records in the buffer that have changed. It makes an internal list of the records changed since the buffer was read in and offers to let you edit them. Use `c-.` (Next Possibility) to move to subsequent records. A numeric argument of 1 means list records changed since the file was last read (this is the default). A numeric argument of 2 means list records changed since the file was last saved.

#### Move Records Among Buffers

Moves records from one buffer to another. A three-column menu pops up. The right column contains the list of Symbolics Concordia buffers read into the editor. You click on two buffers to move records between them. The records in those buffers are then listed in order in the first and second columns.

You move a record name from one column to another by clicking on it with the mouse. The name appears in the same relative position in the other column. To reorder within the other column as you move it, click and hold down the mouse button; when the name appears in the other column, drag the name up or down the column. Release the mouse button when the record is positioned. When you are satisfied, you click on Done. Otherwise, terminate the command with Abort.

This command is available as `m-x Move Records Among Buffers` and as **Move Records Among Buffers** under **Records** in the menu.

#### Query Change Environments

Changes occurrences of one environment to another. You are prompted for an environment name to change, and for one with which to replace it. Confirm your choice by typing one of:

<code>Y</code> or <code>SPACE</code>	Replaces the first environment and moves to its next occurrence.
<code>N</code> or <code>RUBOUT</code>	Skips this occurrence and moves to the next one.
<code>c-L</code> , <code>REFRESH</code>	Redisplays the screen.

ABORT Terminates the command

Any other character terminates the command. This command is not linked to the Undo facility.

#### Remove Record From Buffer

Deletes the record pointed to by the cursor from the buffer.

Links to and from a "removed" record are not disturbed. Therefore, before removing a record, use Show Links to Record to locate and edit or delete the links.

Remove Record From Buffer does *not* remove a record from the record registry. The record is still available for insertion into another buffer with Insert Record. Use of these two commands provides a quick way to move a record from one buffer to another.

You will probably prefer to use the "Move Records Among Buffers" command if you are working with a more than a few records.

#### Rename Record

Renames the record at the cursor position. Prompts for a new name. Symbolics Concordia automatically updates the record header and trailer, all links to the record, and the online display of the record.

Use Rename Record to correct punctuation (except the em dash) or spelling or to change the name of a record. To change the capitalization or character styles used in a record name or to insert an em dash, add a Record-Name field to the record using "Add Record Field" .

If you have a renamed record and a version of that record with its old name is read into your machine (perhaps because you need to refer to an old version of a file for some reason), the record appears to revert to its old name. This can be very confusing, but it is only a local confusion in your environment. You can straighten it out by using Rename Record again. (It is not necessary to patch this re-naming, since it is only a local phenomenon in your environment, and does not affect any other users.) For more information, see the section "Patching a Documentation System", page 442.

**Warning:** You cannot use Rename Record to change the name of a Lisp object record. If, for example, you wish to rename because of a typing error in the record name, you must kill the existing record and create a new record with the correct name. Note that you can save any contents of the old record on the kill ring, before killing the record.

This command is available as **Rename** under **Records** in the menu or as  $m-R$  **Rename Record**.

**Reorder Records** Rearranges the order of records in the current buffer. A menu pops up of all records in their current order. You click and hold the mouse down on a record name. This highlights the name. Then you drag the mouse up or down the menu to transpose the position of the highlighted name with another name. Once the record names are arranged satisfactorily, you can click on **Done** to have the records in the buffer reordered accordingly. Clicking on **Abort** keeps the records in their original order.

This command is available as  $m-R$  **Reorder Records** and as **Reorder Records** under **Records** in the menu.

**Reverse Find Link** Prompts for a documentation topic and for a link view (include, contents, crossreference, precis, or all views) and does a reverse search for a link of that name and view. If it finds a matching link, it positions the cursor at the link.

See also: "Find Link"

**Reverse Find Markup**

Prompts for a markup environment (for example, Commentary, Figure, Enumerate). It then searches for a markup environment of that type. If it finds one, it positions the cursor on the environment diagram line for the environment. Both the beginning and ending environment diagram lines are found by this command.

**Reverse Find Markup String**

Prompts for a string and searches the structural markup and formatting markup in the buffer for any markup containing the string. If it finds the string, it positions the cursor on the beginning of the markup line. Repeating  $s-R$  and  $s-S$  finds additional instances of the string.

$s-R$  maintains its own string history and offers you the last string used by Find Markup String. To repeat the previous search, you just accept this default, that is, press  $s-R$  followed by RETURN.

This command is available as  $s-R$ .

**Set Buffer Disposition**

Lets you lock the file associated with the buffer for exclusive use, or disconnect the buffer from the file. A file cannot be modified by other users when you have locked it.

This command takes as its value one of the arguments: Lock or Disconnect.

Lock	Locks the file for modifications.
Disconnect	Makes the buffer modifiable without locking the file for exclusive use. You must lock the buffer before saving it if you want to save it with its original name.

#### Set Record Name Prefix

Sets the record name prefix prepended to newly created records. Leading white space is ignored.

This command is especially useful for creating descriptive record names when you are converting flat text documents to Symbolics Concordia. See also the "Clear Record Name Prefix Command".

#### Set Record Name Suffix

Sets the record name suffix appended to newly created records. Leading white space is ignored.

This command is especially useful for creating descriptive record names when you are converting flat text documents to Symbolics Concordia. See also the "Clear Record Name Suffix Command" .

#### Show Links from Record

Lists links *from* the specified record *to* other records. It prompts for a record name; the default is the current record. The links are listed by type (include, crossreference, and so on).

This command is available as  $\text{M-X}$  Show Links from Record and as Show Links from Record under **Links** in the menu.

#### Show Links to Record

Lists the links *to* a specified record. It prompts for a record name; the default is the current record. The links are listed by type (include, crossreference, and so on).

This command is available as  $\text{M-X}$  Show Links to Record and as Show Links to Record under **Links** in the menu.

#### Show Outline

Lists the include links for a record and all the include links in its "included" records. It prompts for a topic name; the default is the current record.

Invoking the command on the *top-level record* (the record that, when displayed, shows all the records in your document) shows you the table of contents for a document.

This command is available as `m-X` Show Outline and as Show Outline under **Topics** in the menu.

#### Sort Links in Record

Sorts the links in a record alphabetically by topic name.





## 26. Symbolics Concordia Reference Card

<i>Command</i>	<i>Meaning</i>
c-U s-P	Parses, installs, and prints the current record.
s-.	Edit Record. Prompts for a record topic. <b>Records</b> [Edit] in the menu. m-X Edit Record.
s-(	Beginning of environment. <b>Markup</b> [Beginning] in the menu.
s-)	End of environment. <b>Markup</b> [End] in the menu.
s-^	Remove markup only (and leave the text). <b>Markup</b> [Remove Markup] in the menu.
s->	Moves text on this line against the right margin.
s-	Inserts an em dash (—).
s-=	Centers text between tabstops.
s-A	Beginning of current record. <b>Records</b> [Beginning] in the menu.
s-B	Beginning of environment. <b>Markup</b> [Beginning] in the menu.
s-E	End of current record. <b>Records</b> [End] in the menu.
s-F	Forward environment. Moves to the next markup environment.
s-H	Marks the current record. <b>Records</b> [Mark] in the menu.
s-K	Kills the environment (including text). <b>Markup</b> [Kill] in the menu.
s-L	Creates a Language Form (Lisp) around the Lisp object the cursor is over. <b>Markup</b> [Make Language Form] in the menu.
s-M	Create Markup. Prompts for the name of the markup. <b>Markup</b> [Create] in the menu.
s-P	Previews (parses, installs and displays on the screen) the current record. <b>Records</b> [Preview] in the menu.
s-R	Reverse Find Markup String.
s-S	Find Markup String. Locates a string in any markup, structural or formatting.
s-W	What Record am I?
s-BACKSPACE	Repeats the last minibuffer command, without confirming arguments.
s-HELP	Displays a list of all the Concordia commands.
s-SELECT	Followed by E, P, G or B. Selects an activity (within Concordia).

- `s-SPACE`      Repeats the last minibuffer command, without confirming arguments.
- `s-TAB`         Inserts a tab-to-tabstop.

## **PART V.**

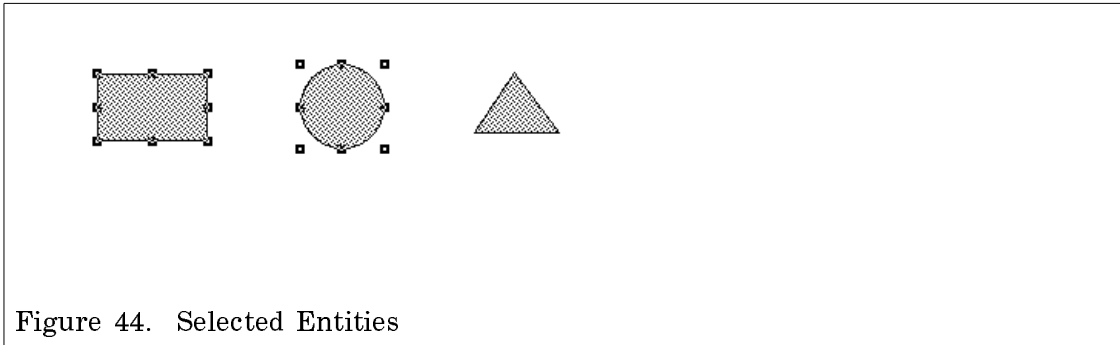
### **USING THE GRAPHIC EDITOR**



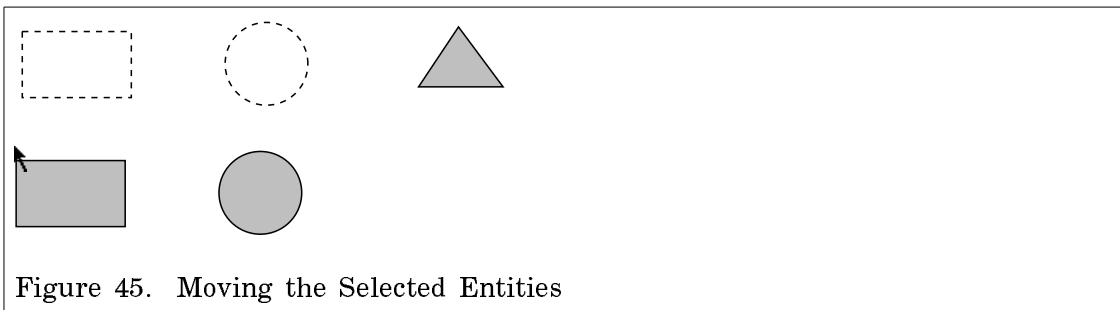
## 27. Overview

The Graphic Editor is an object-based drawing program that lets you create illustrations, diagrams, and other graphics. You construct drawings from shapes, available on a menu. You can include bitmap images picked up from the screen, drawn by Lisp code, or generated by other drawing programs in your pictures.

An object in a drawing (a shape, a grouped set of shapes, or an image) is referred to as an *entity*. Entities are the smallest units on which you can operate. You perform operations on the *selected* entity or entities. Generally, selected entities are visibly marked as selected by the presence of little boxes, called *handles*.



In Figure 44, the rectangle and the circle are selected. If you were to select the Move operation at this point, the rectangle and the circle would be *picked up* and could be relocated using the mouse. You can optionally define and display a grid for the more precise placement of graphic entities.



The interface is very mouse-oriented. Most operations are selected from a menu and performed by mouse clicks. Most of the commands also have keyboard equivalents.


You can get documentation for commands and shapes by clicking **Middle** on them in the menu or the display produced by pressing **HELP**.

Drawings created with the Graphic Editor can be placed in any editor buffer. You can use them as illustrations in Symbolics Concordia documents and put them in mail messages or in a Zmacs text buffer.

If you are editing a previously created picture and want to change the appearance of one of its entities, you select the entity you want to change and then click on **Change** in the menu. This displays a menu of the parameters appropriate to the selected entity. You can also click **Right** on the entity you want to change and select **[Change]** from the menu there, or you can specify **[Change]** directly by clicking **c-m-Right** on the entity when the mouse is over it.

## 28. Getting Started with the Graphic Editor

You can start up the Graphic Editor by itself or from a Symbolics Concordia activity. Both of these Graphic Editors share the same picture/buffer list, and a drawing made in either one can be inserted in a Symbolics Concordia document or any other editor buffer. The Graphic Editor within Symbolics Concordia is the one that Symbolics Concordia uses when you ask to edit a picture by clicking `m-Left` on its representation in a record. Symbolics Concordia uses the current drawing in its Graphic Editor as the default offered when you insert a drawing, but drawings in either Graphic Editor are accessible to Symbolics Concordia. If you are using Symbolics Concordia, it is slightly more convenient to use the Graphic Editor within Symbolics Concordia. If your purpose in using the Graphic Editor is to create a picture to be placed in a mail message or in a Zmacs buffer, it does not matter which Graphic Editor you use.

- To start it by itself, press `SELECT G`. This is the standalone Graphic Editor.
- To start it from inside Symbolics Concordia, click on  in the icon menu in the upper right corner of the screen, or press `s-SELECT G`. This is the Graphic Editor within the Symbolics Concordia activity.

### 28.1. The Graphic Editor Screen Layout

The Graphic Editor window is divided into the following major areas:

- The title in the upper left.
- The main drawing area occupying most of the screen.
- The command menus on the right.
- The command echo area at the bottom.

The Graphic Editor command menus are functionally divided as follows:

Help menu	The Help command.
Undo menu	Commands related to undoing.
I/O menu	Commands related to reading, writing, naming, and hardcopying drawing buffers.
Selection menu	Commands related to selecting and grouping entities.

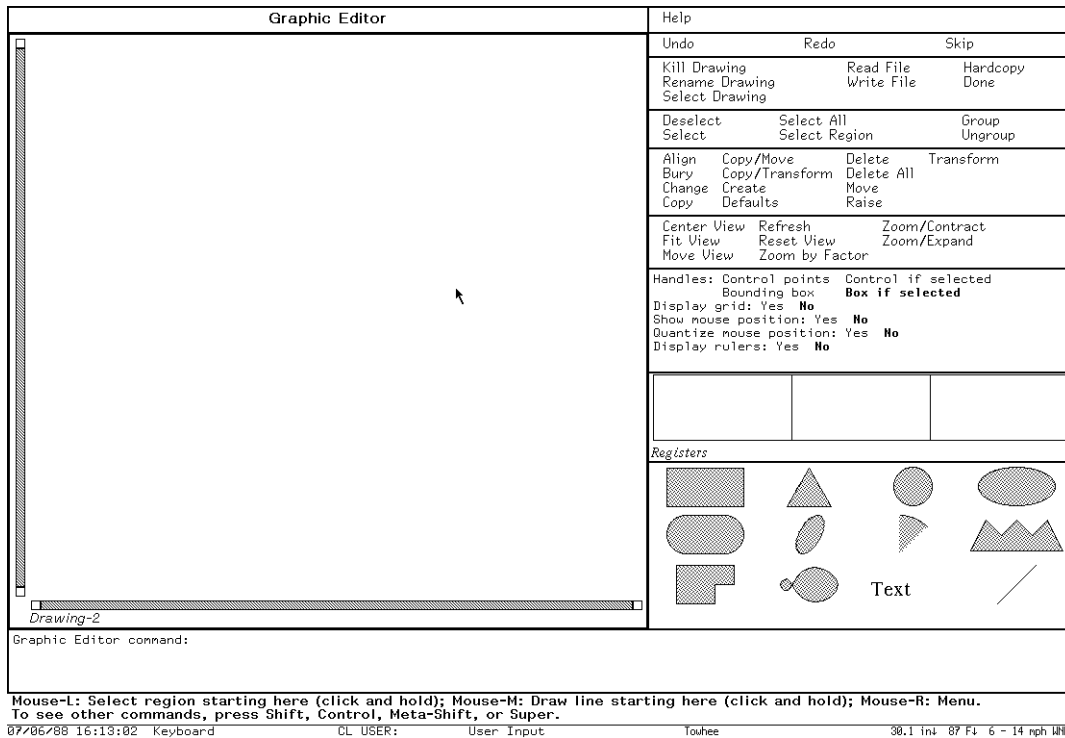


Figure 46. The Graphic Editor Screen

Drawing menu	Commands related to drawing and arranging the entities in the drawing.
View menu	Commands that affect how the drawing is presented in the Graphic Editor only.
Parameter menu	Display and mouse display options.
Registers	Temporary storage areas for saving entities from the drawing or for moving entities from one drawing to another.
Shapes menu	Common entity shapes.

## 28.2. Using the Mouse in the Graphic Editor

You can select most Graphic Editor commands by clicking on them in the command menus that appear at the right of the screen. Clicking-Middle on a command displays help documentation for the command. Clicking-Middle on a shape in the Shapes menu displays documentation for the shape, including how to create it.

You can select some Graphic Editor commands by clicking with the mouse in the Drawing pane. Clicking-Right in the Drawing pane displays a menu of command that you can click on.



Note that in the documentation of Graphic Editor procedures and commands, menu options that you can select by clicking-Left with the mouse appear as [*option name*]. Look in the mouse documentation pane at the bottom of the screen (when a menu option is highlighted) for the meaning of other current mouse gestures.

*Clicks on drawing entities:*

Left	Selects the entity, deselecting whatever was selected before.
Left and hold	Selects the entity, and starts moving it. The entity can be moved until the mouse is released.
⇧-Left	Selects the entity, adding it to whatever was selected before.
⇧-Left and hold	Selects the entity, adding it to whatever was selected before, and starts moving the entire selected set. The set can be moved until the mouse is released.
Left on an entity's handle and hold	Moves that vertex (or analogous piece of non-linear shapes), changing the size and/or shape of the entity.
Middle	Makes a copy of the entity, putting it close to the original.
Middle and hold	Makes a copy of the entity, and starts moving the copy. The copy can be moved until the mouse is released.
Right	Gives a menu of operations on the entity.
⌘-Middle	Copies the entity into the leftmost register.

*Clicks on the blank area:*

Left	Deselects what is currently selected.
Left and hold	Selects whatever is inside the rectangle that is drawn until the mouse is released, deselecting what was selected before.
⇧-Left and hold	Selects whatever is inside the rectangle that is drawn until the mouse is released, adding it to whatever was selected before.
Middle and hold	Draws a line (or an arrow) starting where the mouse is clicked and ending where it is released.
⌘-Left and hold	Creates another entity of the same kind as the last one drawn.

You can also use the mouse to draw and edit Graphic Editor curves and lines. For more information, see the section "Creating Shapes with the Graphic Editor", page 249.

## 28.3. Changing Graphic Editor Defaults

You can change the default appearance of graphic entities and of the Graphic Editor window itself using:

- "Edit Defaults Graphic Editor Command"
- "Edit Interaction Style Graphic Editor Command"
- "Edit Ruler Graphic Editor Command"
- Drawing Parameters Menu Options

### Edit Defaults

[Defaults] Changes the default drawing parameters to be used for new entities. These defaults will all apply to all newly created drawing entities.

To change the appearance or presentation parameters of existing entities use [Change].

[Defaults] pops up a menu, which looks like this:

```

Polygon roundedness: Unrounded Rounded Smoothed
Text character style: NIL.NIL.NIL
Arrowhead(s): Start End
Presentation type: None Documentation Topic Lisp Form Pathname Command

Outline drawing defaults:
Draw outline: Not drawn Opaque Non-opaque
Thickness: 1
Dashed: Yes No

Inside drawing defaults:
Fill inside: Not drawn Opaque Non-opaque
Fill pattern: Black White Gray Patterned
Fill pattern gray level: .25

Other command defaults:
Zoom factor: 4
Hardcopy if too large: Clip Scale Multiple-Pages
Hardcopy orientation: Landscape Portrait
Rotation angle: 90

Other drawing defaults:
Draw faster rather than more accurately: Yes No
<REORT> aborts, <END> uses these values

Sample thicknesses:
1 _____ 2 _____ 4 _____ 8 _____

Sample gray levels:
.75 .50 .33 .25 .12 .10 .09 .08 .06 .05

```

The sample shapes in the shapes menu at the bottom right of the window are displayed using the defaults currently in force.

See also "Edit Interaction Style Graphic Editor Command" and "Edit Ruler Graphic Editor Command" for more details on customizing the Graphic Editor.

**Polygon roundness** How the corners of polygon entities are drawn. The options are Unrounded, Rounded, or Smoothed.

**Text character style**

The character style of text entities. Character styles are in the form family.face.size (for example, FIX.ROMAN.NORMAL). For more information, see the section "What is a Character Style?" in *General User's Guide*.

**Arrowhead(s)** How to draw arrows. You can put arrowheads on either the Start, End, or both ends of lines. When you select either of these arrowhead options, you can also choose to specify the dimensions of the arrowheads.

**Presentation Type** Determines where and when a graphic entity is mouse-sensitive and what kinds of presentation objects are valid. For more information, see the section "Presentation Types for Mouse Sensitive Text and Drawings", page 247.

**Presentation Object** Names a specific presentation object and controls what happens when the entity is clicked on.

### Outline Drawing Defaults

**Draw outline** Outlines for shapes entities can be Not drawn, Opaque, or Non-opaque.

**Thickness** The line thickness of drawn shapes outlines.

**Dashed** You can choose to draw shapes outlines with dashed or undashed lines.

**Line pattern** You can choose Black, White, Gray, or Patterned. Note that these options only appear in the AVV menu when you change the outline line Thickness value.

**Line end shape** You can choose Butt, Square, or Round. Note that these options only appear in the AVV

menu when you change the outline line Thickness value.

**Line joint shape** You can choose Mitre, Bevel, Round, or None. Note that these options only appear in the AVV menu when you change the outline line Thickness value.

### Inside Drawing Defaults

**Fill inside** You can choose how to fill shape entities (Not drawn, Opaque, Non-opaque). If you choose Opaque, shapes entities will hide any entities "behind" them. Non-opaque entities are "transparent".

**Fill pattern** You can fill shapes with color (Black, White, or shades of Gray), or with a Pattern. A menu of sample gray patterns for you to choose from appears at the bottom of the Defaults menu. Gray .25 is the default in this menu. If you choose Patterned, a sample menu of patterns appears (you can click on one of the sample patterns to select it.)

Note that the shapes menu shows the current inside drawing defaults.

### Other Command Defaults

**Zoom factor** The default scale factor for the "Zoom by Factor Graphic Editor Command".

**Hardcopy if too large** How to print drawings that are too large to fit on a page. You can select Clip, Scale, or Multi-pages.

**Hardcopy orientation** You can choose to print drawings Landscape, or Portrait.

**Rotation angle** This is the default  $n$  for the [Rotate  $n$  Degrees] Interactive Transform Type option of the "Transform Entity Graphic Editor Command".

### Other Drawing Defaults

**Draw faster rather than more accurately** You can choose to speed up the display of drawings (at the expense of accuracy).

## Edit Interaction Style

Changes the default user interface to some drawing operations. When you enter this command, this menu of options is displayed:

```

Erase before moving: Don't erase Erase Partially-Erase
Moving feedback: Image Outline
Select origin for rotate, scale, etc.: Choose Center Corner
Text entry mode: AVV menu Line of text
Circular arc input: Center and two radii Two endpoints and bow
Default highlighting: Outline Bounding-Box
Some command options defaults are sticky: Yes No
<REORT> aborts, <END> uses these values

```

### Erase before moving

Whether an object that is being moved should be erased. The choices are Don't Erase, Erase, and Partially Erase. The default is Erase.

### Moving feedback

What the visual feedback during the moving operation should be. The choices are Image and Outline. The default is Image.

### Select origin

What the center of rotation should be. The choices are Choose, Center, and Corner. The default is Choose.

### Text entry mode

How text is created. The choices are from an AVV menu or line of text from the minibuffer. The default is from the minibuffer.

### Circular arc input

How arcs are created. The choices are [Center and two radii] and [Two endpoints and bow]. The default is Two endpoints and bow. For more information, see "The Circular Arc/Sector Graphic Editor Shape".

### Default highlighting

How entities are highlighted when the mouse is over them. The choices are Outline and Bounding Box. The default is Outline.

### Sticky defaulting

Whether arguments for some command options should have sticky defaults, that is, once you specify an option it becomes the default for the next use of that command. The default is No.

To change the appearance of existing entities, use **Change**. To change the appearance of entities you subsequently create, use **Defaults**.

### Edit Ruler

Allows changing how the ruler is displayed. When you enter this command, this menu of options is displayed:

```
Scale unit: Centimeter Inch Pixel
Display rulers: Yes No
Units per major division: 1
Units per minor division: 1/10
Grid size: 1
Display hardcopy box: Yes No
<REORT> aborts, <END> uses these values
```

**Scale unit**            The units used, centimeters, inches, or pixels.

**Display rulers**        Specify whether the rulers are displayed.

**Units per major division**  
Ruler scale increment for labeled units (for example 1 to label each whole unit on the ruler).

**Units per minor division**  
Ruler scale increment for ticks between units (for example .1 for 10 tick marks between each unit on the ruler).

**Grid size**            The grid size (when the grid is displayed).

**Display hardcopy box**  
Whether the hardcopy box is displayed. The hardcopy box shows the page boundaries on the drawing pane.

### Drawing Parameters

You can change some display options by clicking in the Drawing Parameters pane (which is located just above the Registers pane to the right of the Drawing pane):

```
Handles: Control points    Control if selected
         Bounding box     Box if selected
Display grid: Yes No
Show mouse position: Yes No
Quantize mouse position: Yes No
Display rulers: Yes No
```

Handles	You can choose to display either the bounding box or control points for only selected or for all Graphic Editor entities.
Display grid	A grid is useful for the precise placement of drawing entities. You can specify the grid size (in the default units). Note that smaller grids take longer to display. You can change the default grid units with the "Edit Ruler Graphic Editor Command".
Show mouse position	Shows the drawing window coordinates of the mouse cursor position (in the default units). You can change the default units with the "Edit Ruler Graphic Editor Command". You can use these values in Graphic Editor commands to explicitly locate and define drawing entities.
Quantize mouse position	When you select this option, mouse clicks in the drawing pane "snap to" the nearest grid point. A "o" marks the actual cursor position; a "+" marks the grid point that the cursor location "snaps to".
Display rulers	Displays x and y axis drawing window rulers. You can use "Edit Ruler Graphic Editor Command" to change the default ruler tic units. Note that drawing window coordinates will be displayed in these units.

## 28.4. Controlling Mouse-Sensitivity in Graphic Editor Drawings

You can assign mouse-sensitivity to individual graphic entities or grouped entities in Graphic Editor drawings by associating them with a Presentation Type and a Presentation Object. You can set the default Presentation Type for all new graphic entities in the [Defaults] menu ("Edit Defaults Graphic Editor Command"). You can use the [Change] menu to set the Presentation Type and Object for selected graphic entities, or to override any defaults ("Change Entity Graphic Editor Command").

The Presentation Type determines where and when a graphic entity is mouse-sensitive and what kinds of presentation objects are valid. You can choose from the following Presentation Types:

None   Documentation Topic   Lisp Form   Pathname   Command

For more information, see the section "Presentation Types for Mouse Sensitive Text and Drawings", page 247.

The Presentation Object names a specific presentation object and controls what happens when the entity is clicked on.

For example, if you set the Presentation Type of an entity to Documentation Topic, the entity will be sensitive whenever a documentation topic would be (that is, inside the Document Examiner, for instance). If the Presentation Object is the topic "Controlling Mouse Sensitivity in Graphic Editor Drawings", the entity will behave as a crossreference to this topic.

You can set the Presentation Type of selected graphic entities using the "Change Entity Graphic Editor Command". You can set the default Presentation Type of all new graphic entities using the "Edit Defaults Graphic Editor Command".

### Using the Change Menu to Set Presentation Parameters

When you click on [Change] for a selected entity, the following menu of options appears:

```
Entity options:
Presentation type: None Documentation Topic Lisp Form Pathname Command
Roundedness: Unrounded Rounded Smoothed

Outline drawing options:
Draw outline: Not drawn Opaque Non-opaque
Thickness: 1
Dashed: Yes No

Inside drawing options:
Fill inside: Not drawn Opaque Non-opaque
Fill pattern: Black White Gray Patterned
Fill pattern gray level: .25
```

You can specify a Presentation Type and a Presentation Object for the entity.

When you choose a Presentation Type other than None, additional Presentation Object prompts will appear.

### Using the Defaults Menu to Set Default Presentation Parameters

When you click on [Defaults] for a selected entity (or when you enter the "Edit Defaults Graphic Editor Command"), the following menu of options appears:



```

Polygon roundedness: Unrounded Rounded Smoothed
Text character style: NIL.NIL.NIL
Arrowhead(s): Start End
Presentation type: None Documentation Topic Lisp Form Pathname Command

Outline drawing defaults:
Draw outline: Not drawn Opaque Non-opaque
Thickness: 1
Dashed: Yes No

Inside drawing defaults:
Fill inside: Not drawn Opaque Non-opaque
Fill pattern: Black White Gray Patterned
Fill pattern gray level: .25

Other command defaults:
Zoom factor: 4
Hardcopy if too large: Clip Scale Multiple-Pages
Hardcopy orientation: Landscape Portrait
Rotation angle: 90

Other drawing defaults:
Draw faster rather than more accurately: Yes No

```

When you choose a Presentation Type other than None in the [Defaults] menu, you are prompted for a presentation object of the appropriate kind whenever you create an object.

## Presentation Types for Mouse Sensitive Text and Drawings

### *Presentation Type:*

- |                     |  |
|---------------------|--|
| None                | This object is not mouse sensitive.  |
| Documentation Topic | This object is sensitive as a crossreference. You are prompted for the name of a documentation record. The documentation record must be available when the drawing is being displayed (otherwise an error message is displayed).   |
| Lisp Form           | This object is sensitive as a Lisp expression. The sensitivity is apparent only when viewed in a context that accepts Lisp expressions. (In other words, it will be sensitive if displayed via Show Documentation in a Lisp Listener, but not when displayed in Document Examiner.)<br><br>This is useful mostly for embedding calls to user application code in a picture that the application uses.      |
| Pathname            | This object is sensitive as a file pathname. The sensitivity is apparent only when viewed in a context that accepts a pathname.<br><br>Note that if you move objects from one site to another, you must make sure that the host is defined for any physical pathname presentation objects. An error will be displayed if you use physical pathnames, and the physical host is not defined at the new site. |
| Command             | This object is sensitive as a command. The sensitivity is apparent only when viewed in a context that accepts a command.   |

When this is selected, you are prompted for a command table and a command. Most system commands are in the Global command table. Application-specific commands are usually in a command table that has the same name as the program name used in the application's definition.

See the function **dw:define-program-framework** in *Programming the User Interface*, and see the function **dw:define-program-command** in *User Interface Dictionary*.

Of these types, the one that is most generally useful for straight documentation is Documentation Topic. Using this presentation type, you can embed crossreferences directly into your illustrations.

The other presentation types are mostly for embedding a Graphic Editor drawing into a software application. There are three ways you can do this.

- Your application can call (**cp:execute-command** "Show Documentation" *your topic*), where *your topic* contains mouse-sensitive drawings.
- Your application can call (**graphic-editor:draw-drawing** *your-drawing*).
- You can use `M-X` "Insert Graphic Editor Drawing Code" in a Lisp buffer to write a function for drawing your drawing that your application can call. Note that the drawing function produced this way should be called within **graphics:with-room-for-graphics**.

For more information about linking Symbolics Concordia to your software applications, see the section "Application Programmers' Interface Guide to Symbolics Concordia", page 458.

## 29. Creating Shapes with the Graphic Editor

the "Create Entity Graphic Editor Command" creates graphic entity shapes. You can create most shapes by clicking on the shape in the Shapes menu in the lower right-hand corner of the window.

The following shapes are not in the Shapes menu. You can create these shapes by clicking Right on [Create] in the Drawing command menu and clicking Left on the shape in the shapes menu.

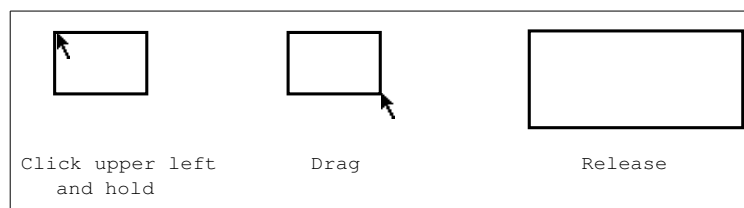
- Cubic Spline
- Be'zier Cubic
- Quarter Ellipse

These are the shapes that you can select using the Graphic Editor shapes menu:



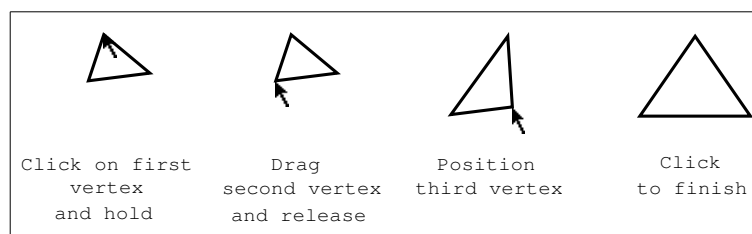
Rectangle

Click Left to position the upper left corner of the rectangle and hold the mouse down while moving the lower right corner.



Triangle

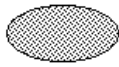
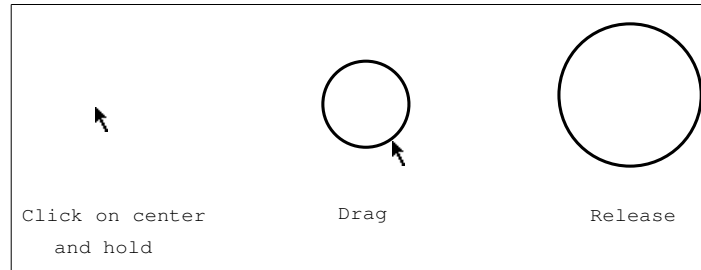
Click Left on the first vertex and hold the mouse down and release at the second vertex. Then click on the third vertex.





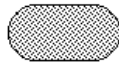
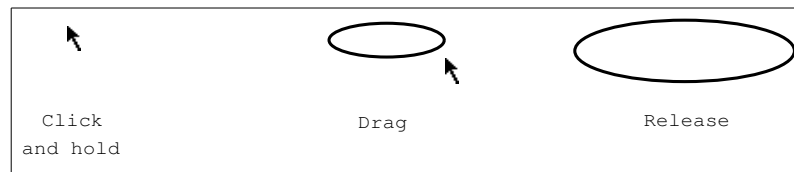
### Circle

Click Left on the center and hold the mouse down and release at any point on the circumference.



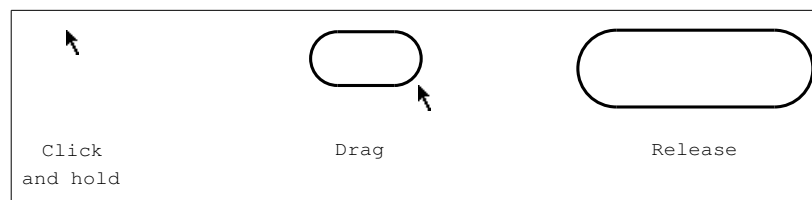
### Rectilinear Ellipse

Click Left on the upper left corner of the bounding box and release at the lower right corner of the bounding box.



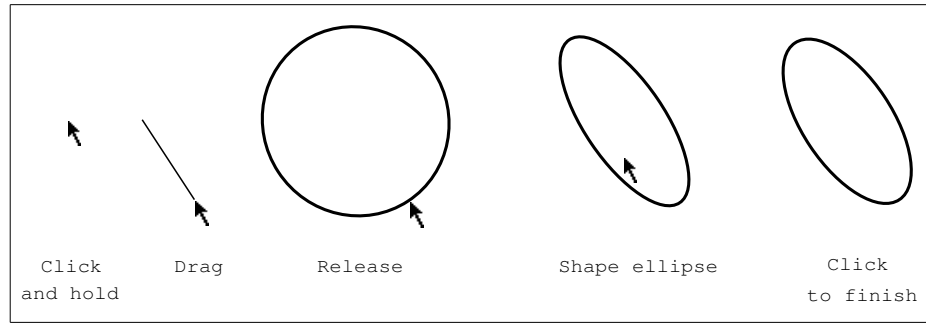
### Oval

Click Left on the upper left corner of the bounding box and release at the lower right corner of the bounding box.



### Ellipse

Click Left on the center point, and hold down and release at the end of one semi-axis. Click on a point on the circumference.

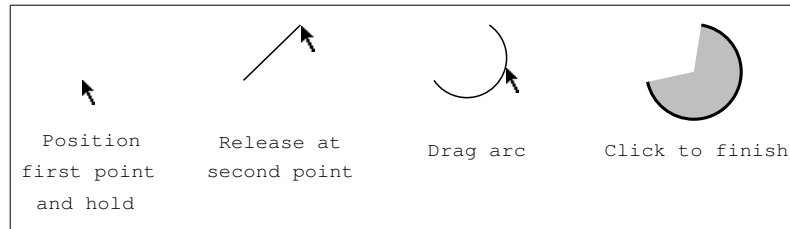


**Circular Arc/Sector**

There are two methods you can use to define arc entities in the Graphic Editor:

- Two endpoints and bow
- Center and two radii

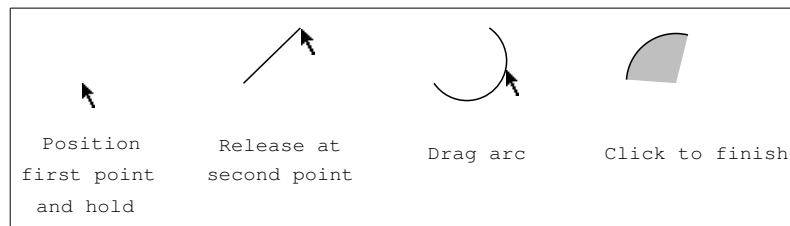
The default method is to define two endpoints and a bow. Click Left on one end of the arc and hold. Drag and release at the other end of the arc. Sweep the mouse and release it to shape the arc. Click Left to finish.



You can change the way arcs are created by clicking on the following option in the the "Edit Interaction Style Graphic Editor Command" AVV menu.

Circular arc input: **Center and two radii** Two endpoints and bow

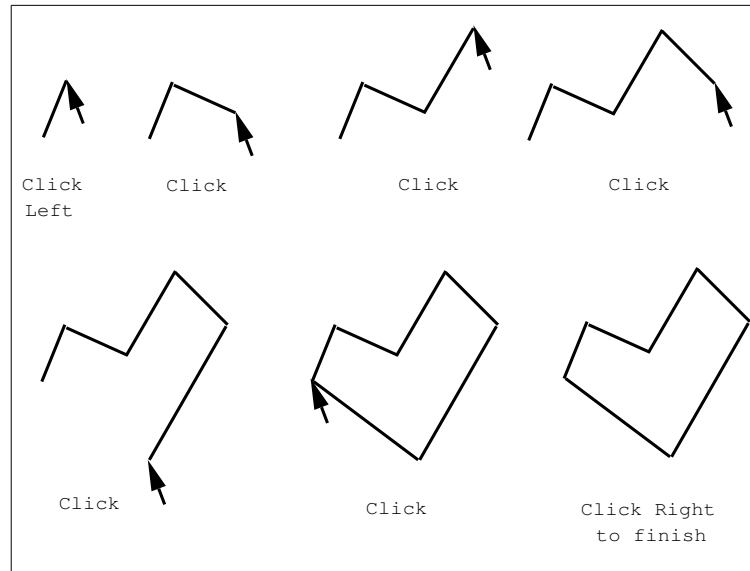
When you create an arc using this option, click on the center of the arc, drag the cursor and click to define the first radius, sweep the cursor and click to define the second radius.





### Polygon

Click on successive points.



When you are drawing a shape with the mouse, clicks are interpreted as follows:

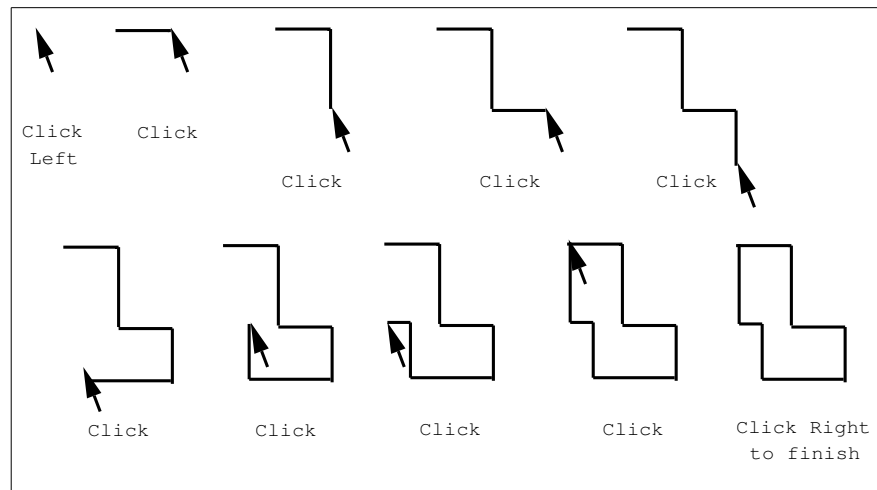
Left	Add this point. You can click Left and hold to draw a freehand curve.
⇨-Left	Add this point and terminate the set.
Middle	Add the nearest point already in the set. This is useful for drawing closed polygons accurately.
⇨-Middle	Add the nearest point and terminate the set.
Right	Terminate the set without adding any new points.



### Rectilinear Lines

Click on successive points. Each new line will be limited in the direction it can move from the previous point to ensure that all angles are right angles.

When you are drawing a shape with the mouse, clicks are interpreted as follows:

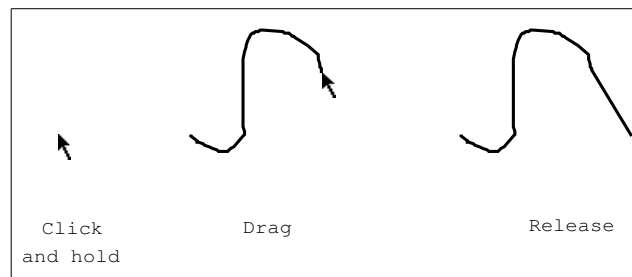


Left	Add this point. You can click Left and hold to draw a freehand curve.
⇨-Left	Add this point and terminate the set.
Middle	Add the nearest point already in the set. This is useful for drawing closed polygons accurately.
⇨-Middle	Add the nearest point and terminate the set.
Right	Terminate the set without adding any new points.



### Freehand Curve

Click on the first point and draw while holding down the mouse.



When you are drawing a shape with the mouse, clicks are interpreted as follows:

Left	Add this point. You can click Left and hold to draw a freehand curve.
------	---

<code>=h-Left</code>	Add this point and terminate the set.
Middle	Add the nearest point already in the set. This is useful for drawing closed polygons accurately.
<code>=h-Middle</code>	Add the nearest point and terminate the set.
Right	Terminate the set without adding any new points.

## Text Text

You can enter text entities by typing them in the minibuffer (Line of Text mode) or by entering text entity options in the text entry AVV menu (AVV Menu mode).

You can use the "Edit Interaction Style Graphic Editor Command" to select the Text entry mode. (Line of Text is the default).

- **Line of Text mode**

Enter text strings by typing in the minibuffer. The default character style is `NIL.NIL.NIL` (which translates to `FIX.ROMAN.NORMAL`). Pressing `TAB` after you type the string prompts you for character style. Click Left to position the string in the drawing.

- **AVV Menu mode**

When you create a text entity an AVV menu appears. You can use the menu to define the text string and to specify how [Transform] operations on the entity will change its appearance.

```
String: some text
Character style: NIL.NIL.NIL
When scaled: Same font Different size font
When rotated: Just start moves Slope changes Stretches horizontally
                Tilts and stretches
<ABORT> aborts, <END> uses these values
```

String            The string

Character style

`NIL.NIL.NIL` (the default) translates to `FIX.ROMAN.NORMAL`

When scaled    Specifies how the appearance of the text string changes when you use [Trans-



form] to change the size of the entity. There are two mutually exclusive options;

Same font  
Different size font

With Same font the characters look the same, but their placement changes when scaled and/or rotated.

With Different size font the size of the characters changes when the text entity is scaled (and their orientation changes when rotated).

When rotated Specifies how the text string appearance changes when you use [Transform] to rotate the entity. The options you can select depend on which When scaled options you chose.

You can press `c-m-Right` to change the following characteristics of a selected text entity.

---

```
Change Text-47:

Entity options:
String: Some Text
Character Style: NIL.NIL.NIL
When scaled: Same font Different size font
When rotated: Just start moves Orientation changes
                Stretches horizontally Tilts and stretches

Drawing options:
Opaque: Yes No
Fill pattern: Black White Gray Patterned
<BEORT> aborts, <END> uses these values
Click here to fill in from current defaults.
```

---

Figure 47 summarizes the text transformation options.

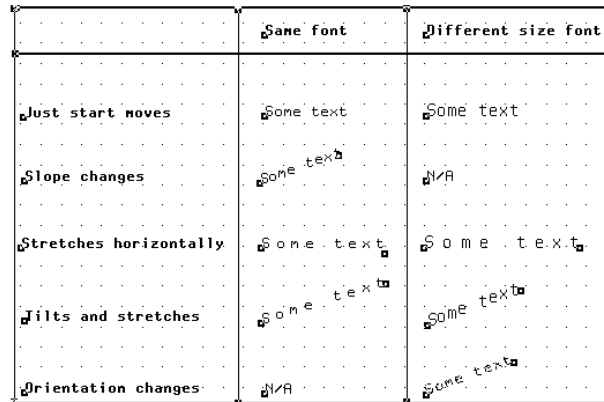


Figure 47. Text Transformation Options

#### Just start moves

You can move the string by clicking on its first character.

#### Stretches horizontally

The string has horizontal whitespace added to make it reach a certain point. Hold down after clicking and release at the other end of the stretch.

**Slope changes** When you rotate the text entity, the orientation of the character glyphs remains the same (see Figure 47).

#### Tilts and stretches

The string is both tilted and stretched.

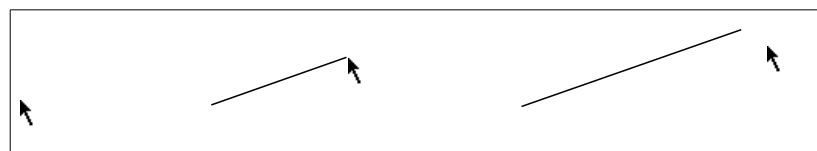
#### Orientation changes

The string is spaced normally, but the characters' baseline is tilted from the horizontal.



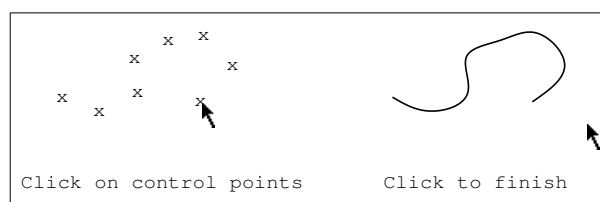
Line

Click on the beginning of the line and drag the mouse to the other end. Release at the end of the line.



The following shapes are selectable only by clicking-Right on [Create] in the Drawing menu or by entering the Create Entity command specifying one of the following entity types from the menu.

**Cubic Spline** Click on successive control points. When the set of points is terminated, they are connected by a smooth cubic spline.



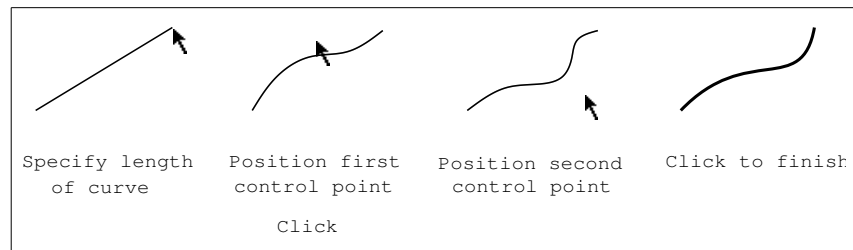
Note that when you are drawing cubic splines, it is often useful to set the Control points option in the defaults menu (just above the Registers to the right of the drawing pane).

Handles: <b>Control points</b>	Control if selected
Bounding box	Box if selected

When you are drawing a shape with the mouse, clicks are interpreted as follows:

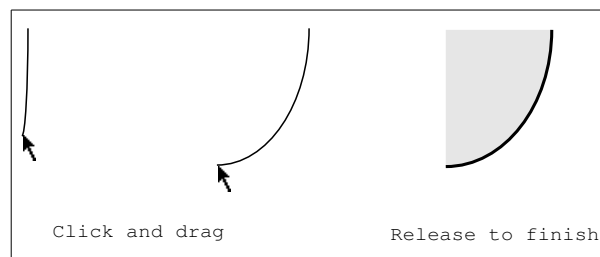
Left	Add this point. You can click Left and hold to draw a freehand curve.
⇨-Left	Add this point and terminate the set.
Middle	Add the nearest point already in the set. This is useful for drawing closed polygons accurately.
⇨-Middle	Add the nearest point and terminate the set.
Right	Terminate the set without adding any new points.

**Be'zier Cubic** Click Left on the first point and hold, release on the final point. Then click on the two intermediate control points.



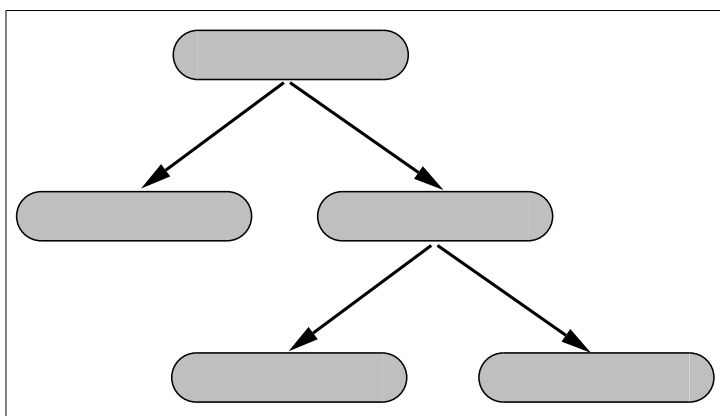
### Quarter Ellipse

Click Left on the starting point, hold and release at the other end.

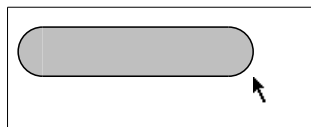


## 30. Creating Drawings with the Graphic Editor

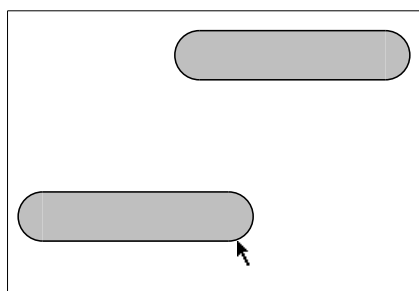
You create drawings by assembling shapes and lines. For example, to create a simple tree structure like this, follow these steps:



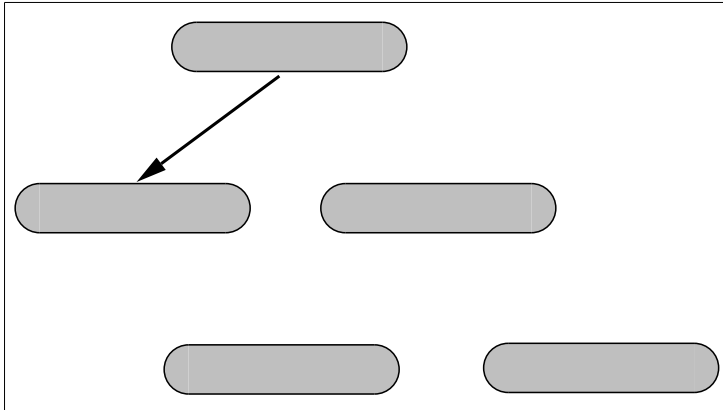
1. You first create one of the ovals.



2. Now, you can copy the first oval to create the other ovals. An entity is selected when you create it, so you can copy your oval by clicking on [Copy] or [Copy/Move] in the menu, or by clicking Middle on the oval. Since you want to position the second oval, you probably want to use Copy/Move or click Middle and hold (which allows you to drag the copy into position). You can repeatedly copy ovals until you have created all five of them.



Now connect them with lines. One way to create several lines with the same characteristics is to use the "Edit Defaults Graphic Editor Command" to change the default appearance of all new lines. (For example, you can change the default line Thickness to 2, and change the Line shape to Arrow.)



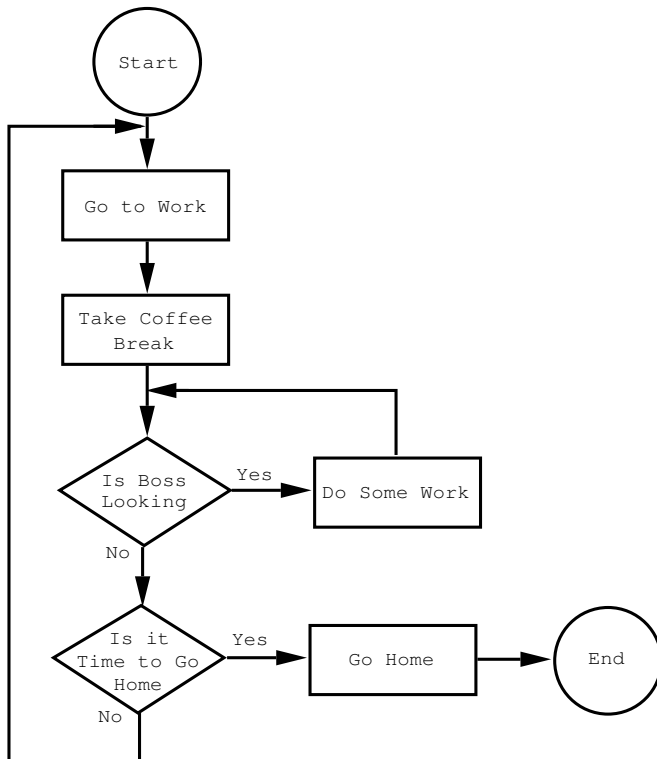
You can also draw lines using the original defaults, select them all and then use [Change] to edit their appearance.

```
Entity options:
Line shape: Line Arrow Double-headed arrow
  Arrow head length: 10
  Arrow base width: 5

Drawing options:
Opaque: Yes No
Thickness: 2
Dashed: Yes No
Line pattern: Black White Gray Patterned
Line end shape: Butt Square Round
Line joint shape: Miter Bevel Round None
<ABORT> aborts, <END> uses these values
```

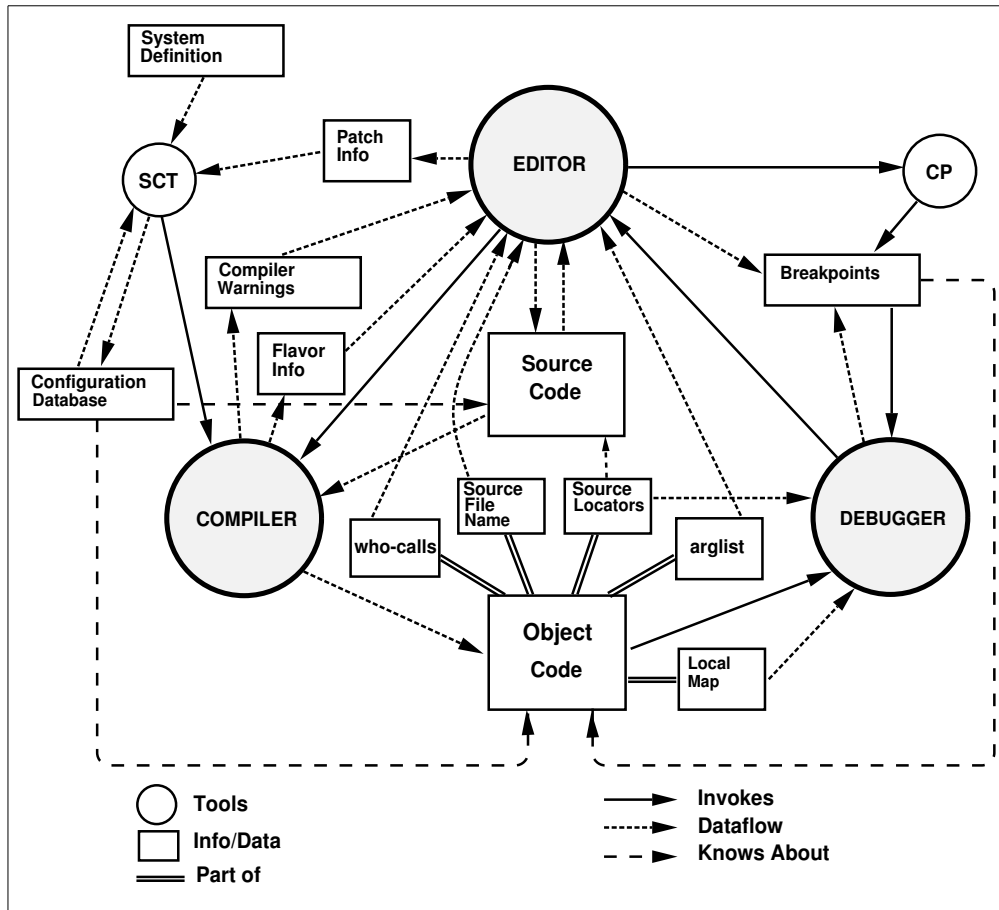
The kind and complexity of drawings that can be made with the Graphic Editor is limited only by your imagination. The remainder of this section gives some examples of drawings done with the Graphic Editor and some of the procedures used to create them. For additional advice and tricks of the trade, see the section "Strategies and Hints for Using the Graphic Editor", page 273.

## Flow Charts and Syntax Diagrams

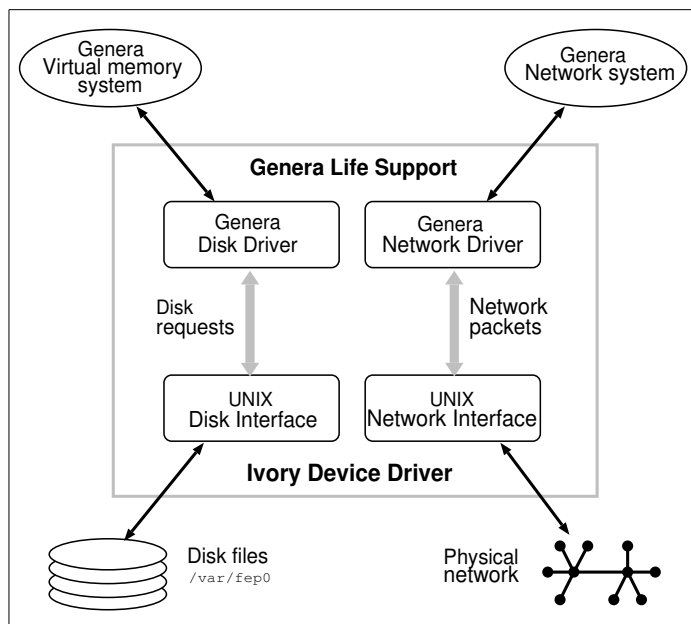


Here elements are shapes from the menu, connected by unfilled rectilinear lines. The arrow heads are short line segments changed to have thickness 2 and arrow heads.

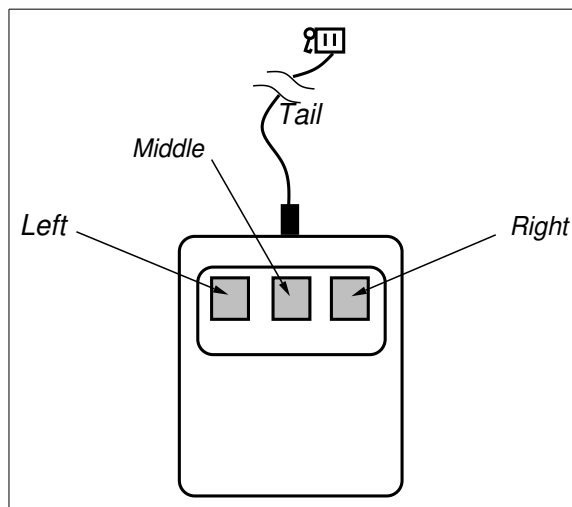
### Complex Structure Diagrams







### Part or Component Diagrams



Simple shapes are combined and the Rounded option is chosen to imitate the shape of the mouse.

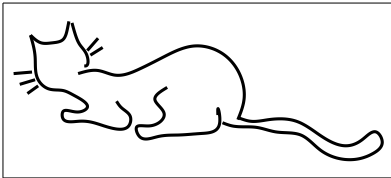
## Screen images

<i>The System Menu</i>		
<i>Windows</i>	<i>This window</i>	<i>Programs</i>
Create	Move	Lisp
Select	Shape	Edit
Split Screen	Expand	Inspect
Layouts	Hardcopy	Mail
Edit Screen	Refresh	Trace
Set Mouse Screen	Bury	Emergency Break
	Kill	Frame-Up
	Reset	Namespace Editor
	Arrest	Hardcopy
	Un-Arrest	File System
	Attributes	Document Examiner

The system menu was captured using FUNCTION 0 Q and placed in a Graphic Editor drawing using the Add Image command. Other graphic entities can be added to provide composite image and graphic editor drawings. (See the section "Using Screen Images as Illustrations", page 322.)

## Illustrations

Cubic splines can be used to create more lifelike pictures.



## 31. Managing Drawings

### 31.1. Changing the View of Drawings

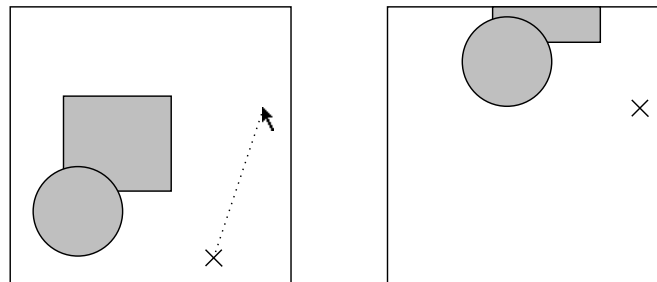
When you start up the Graphic Editor, the drawing pane shows only the lower left corner of an 8.5 by 11 inch piece of paper. Drawings are not limited to the size of the pane (or to 8.5 by 11). You can extend drawings as far in any direction as you want. When you hardcopy a large drawing, you can scale it to fit on an 8.5 by 11 paper, or have it printed on multiple sheets ("Edit Defaults Graphic Editor Command").

For Symbolics Concordia purposes, large drawings should be scaled when inserted in records; see the section "Specifying the Size of Pictures", page 180.

Drawing a large picture requires scrolling around to different parts of the *drawing board*. You can scroll using the scroll bars (see the section "Scrolling with the Mouse" in *Genera Handbook*), or you can use Move View.

Move View

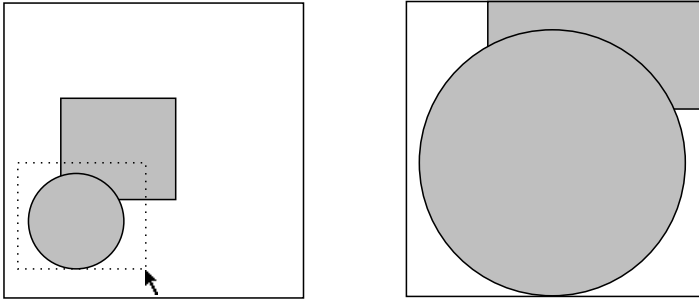
[Move View] Adjusts the view of the drawing or image. The drawing is adjusted so that the first point you click on in the drawing pane is shifted to the second point in the pane that you click on. For example:



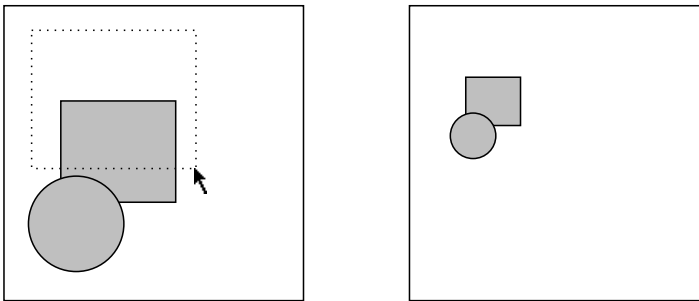
The location of the first click is indicated by the X, the second by the cursor location. The drawing is scrolled up and to the right, as shown (the X is left in the picture for reference, on the screen it disappears).

When you are working on a large drawing, you can lose track of the "big picture". The Fit View command adjusts the display of the entire drawing to fit in the drawing pane. This adjustment is for display purposes only and does not affect the actual size of entities in the drawing or image.

The Zoom commands allow you to expand and contract the display of sections of the drawing so that you can see more or less of the entire drawing. For example, Zoom/Expand allows you to specify a section of the drawing to have fill the entire pane:



Zoom/Contract allows you to specify a section of the drawing pane into which to place the entire current view, exposing more of the surrounding drawing:



These commands, like Fit View, effect only the display, not the actual size of the drawing. Use the [Scale] option of the "Transform Entity Graphic Editor Command" to change the actual size of selected drawing entities.

When you have changed the view of your drawing with any of these commands, you can always get back to your initial state with the Reset View. This is the command to use when you lose your place and want to start out fresh.

## 31.2. Copying Drawings

There are two main ways to copy drawings to other drawings, using the Copy Drawing command, or using registers.

You can copy a Graphic Editor drawing to a bitmap image using the "Copy Drawing to Image Graphic Editor Command".

### Using the Copy Drawing Command to Copy Drawings

Enter the Copy Drawing command by typing the following at the Graphic Editor command prompt. This command makes a copy of an entire drawing. It allows you to make a series of drawings quickly and easily, each just a little modified from the previous.

Copy Drawing TAB

You will be prompted for the name of the source and destination drawings.

### Using Registers to Copy Drawings

Alternatively, you can place a copy of an entity or several entities from a drawing into registers and add them back to the same or different drawings. This allows you to reuse an entity in several drawings, or to make a new drawing containing elements from a previous drawing slightly rearranged.

Place entities in registers by first selecting the entity or entities to be stored, and then clicking on one of the three registers. If you are storing a very large entity in a register, the scaling sometimes causes it to look messed up. When it is retrieved from the register, its display will be correct.

If you need more than three registers to store entities in, use the Store Into Register command and give it the name of a register. Graphic Editor registers are similar to Zmacs registers (see the section "Registers in Zmacs" in *Editing and Mail*). The three visible registers are Register-1, Register-2, and Register-3. You give additional registers any names you like. Since the additional registers are not visible, it helps to give them mnemonic names according to what you have stored in them.

You retrieve the contents of one of the three visible registers by clicking on it or by using Retrieve Register and giving the register name. Pressing HELP displays a list of all the registers you have specified.

## 31.3. Storing Drawings in Files

When you insert a drawing in a Zmacs or Symbolics Concordia buffer, a representation of the picture is actually inserted in the buffer. This is done so that pictures can be used in documentation (or other files) by sites not having the Graphic Editor software. The stored representation is not editable, however. A drawing inserted in a file cannot be edited unless the source for it is still present in the Graphic Editor or is available in a file written from the Graphic Editor.

The drawings loaded in your Graphic Editor are in buffers. These are similar to Zmacs buffers in that they can either be associated with a file or not. However, unlike Zmacs, several drawing buffers can be associated with the same file. When you create a drawing, it is not associated with a file. You can associate it with a file by using the "Write File Graphic Editor Command", or the "Set Drawing File Graphic Editor Command".

There are several commands for file manipulation in the Graphic Editor:

Write File	Writes the current drawing buffer out to a file. Actually, changes the file associated with the drawing to be the given file, and then writes the file to contain all the drawings associated with that file.
------------	---

Read File	Reads in a file full of drawing buffers, then selects one of the drawings in the file. If there is more than one, you are presented with a menu to choose from.
Set Drawing File	Associates a drawing with a file name.
Show File Drawings	Displays a list of the drawings in the specified file.

You can use Write File to save the current drawing to a file. Note that each time you use Write File, images of all drawings in the file are saved.

To conserve disk space you might want to use the "Set Drawing File Graphic Editor Command" to associate a drawing with a file.

```
Graphic Editor command: Set Drawing File tiny-font  
r:>mydirectory>drawings.bin.newest
```

Then use "Write File Graphic Editor Command" to save all of the drawings associated with the file when you are finished editing all of your drawings.

```
Graphic Editor command: Write Drawing File  
r:>mydirectory>drawings.bin.newest
```

You can use the "Show File Drawings Graphic Editor Command" to display the drawings in a specified file, or you can click Right on [Select Drawing] in the command menu to see a summary of drawings in all of the files that have been read in. Then you can choose what drawing file(s) to write.

The files that the Graphic Editor writes are binary files. You can give them a file extension of .bin, or if you prefer, you can use something that suggests their special purpose as drawing source files, perhaps .pic or .dwg.


If your drawings are being used in documentation that is part of a system, you should add a module to your system declaration to record the source files for drawings. This insures that any operations performed on the entire system (copying it to another file server, or reapp-protecting it, for example) operate on the drawing files, also. See the section "The System Declaration for a Documentation System", page 431.

## 32. Using Drawings in Symbolics Concordia Documents

### 32.1. Inserting a Bitmap Image Into a Graphic Editor Drawing

If you want to use a bitmap image (such as a Symbolics screen image or a scanned in picture from another document) as an illustration in a Symbolics Concordia document, you must first put it into a Graphics Editor drawing.

Use this procedure to insert a bitmap image into a Graphic Editor drawing. For more information, see the section "Bitmap Editor Basic Concepts", page 319.

1. Select the Graphic Editor activity (using `SELECT G` or by clicking on  in the upper-right of the Symbolics Concordia window).
2. Click Left on [Select Drawing] and enter the name of drawing to contain the image (or click Right and select from the menu of active drawings). The drawing name is the name that uniquely identifies the drawing in the Graphic Editor and in the Symbolics Concordia document ( see the section "Create Picture").

Graphic Editor command: `Select Drawing tiny-font`

3. Use the Add Image command (with the named image and a scale factor) to include the bitmap image in the current Graphic Editor drawing. Move the mouse cursor and click Left to position the bitmap image in the drawing pane.

Note that in general it is better to accept the default scale factor (1 for full size). You can use the `PictureScale` environment attribute to resize your drawing when you insert the final picture in your document. (See the section "Specifying the Size of Pictures", page 180.)

Graphic Editor command: `Add Image show-font-display`

The image "show-font-display" is then inserted into the current drawing at the mouse cursor location.

4. Use Graphic Editor commands to add drawing entities (such as arrows and comments) as needed.
5. Click on [Done] to exit the Graphic Editor activity.

## 32.2. Including a Graphic Editor Drawing in a Document

To include a picture (that is, a Graphic Editor drawing) in a Symbolics Concordia document:

1. Select the Symbolics Concordia activity. If you entered the Graphic Editor from a Symbolics Concordia activity, clicking on [Done] in the Graphic Editor returns you to Symbolics Concordia. Otherwise, press `SELECT W` to enter Symbolics Concordia.
2. If the drawing is not loaded, you must first load the file that contains it. To load a drawing in a drawing file enter the Graphic Editor and select [Read File], specifying the pathname of the file that contains the drawing.

If the drawing is already included in an existing Symbolics Concordia record, you can also load the drawing by clicking `m-Left` on the picture icon that references the drawing, and then selecting [Edit Picture]. The file containing the drawing is loaded, and the drawing becomes the active drawing. Note that this procedure also loads all of the drawings in the referenced file.

3. Use Symbolics Concordia buffer search and navigation commands (such as `s-. topic name`, or `c-X c-S string`) to position the cursor to the location in your document where you want to insert the drawing.
4. Enter `m-X Create Picture` and specify the name of a (loaded) Graphic Editor drawing to be inserted (the current drawing is the default). This creates an icon in your document that represents the drawing.

```
🔗 Picture "mine" from file SAP:>sys>graphic-editor>documentation.pic.newest
```

When you display the Symbolics Concordia topic record, the drawing is displayed.

5. You can enclose the picture icon in a Figure environment to add a caption, or to reference the drawing from other places in the document.

```
Figure
🔗Ref, Code horizontal-bug
  shows a bug.
Figure ...
  🔗 Picture "bug-4" from file SCRC|SAP:>sys>
  🔗Caption A Bug
  🔗Tag, Code horizontal-bug
Figure ...
```



6. You can change the size of the drawing in a Symbolics Concordia environment by editing the `PictureScale` attribute of the surrounding environment. Note that `PictureScale: .5` gives a nicely-sized full screen image. For more information, see the section "Specifying the Size of Pictures", page 180

When you display the topic, the picture icon is interpreted, and the drawing containing the bitmap image is displayed. For more information, see the section "Graphic Editor: Putting Pictures in Text", page 179.



## 33. Strategies and Hints for Using the Graphic Editor


### 33.1. Creating Non-Standard Shapes

The shapes in the Shapes menu and the additional entities in the Create menu provide most of the building blocks for charts and diagrams, but pictures of real-life objects require perspective, which means that they cannot be made with standard geometric shapes.

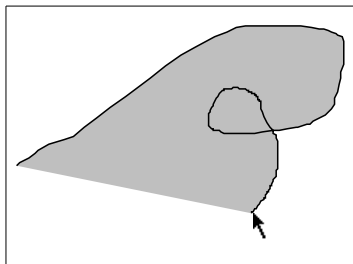
Creating perspective drawings requires some imagination, but there are some general techniques that can help.

#### 33.1.1. Drawing Irregular Shapes

You can draw freehand curves, but the screen resolution and mouse tracking combine to make this unsatisfactory for all but the smallest entities.

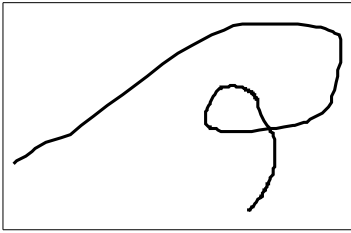
Click on the curve shape, , in the menu and then position the mouse, click Left, hold the button down, and draw with the mouse.

A freehand curve looks like this:



Note that (unless you have changed the drawing defaults), the Graphic Editor fills the new shape with Gray .25. This can be useful when you want to combine shapes. See the section "Combining Entities to Produce Complex Shapes", page 275.)

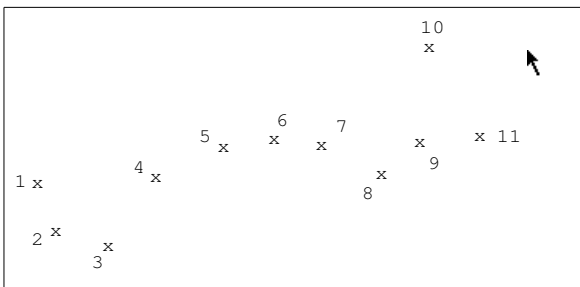
You can use the "Edit Defaults Graphic Editor Command" to change the drawing defaults. For example, changing the Outline Drawing Default to Thickness: 2, and changing the Inside Drawing Default to Fill Inside: Not Drawn, looks like this:



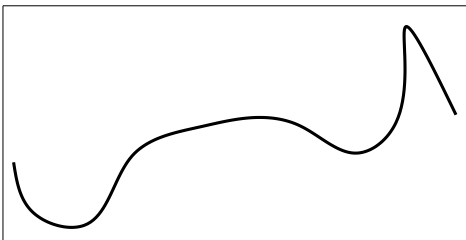
You can select the cubic spline shape to produce smoother shapes. When you select the cubic spline shape, the Graphic Editor draws a smooth curve between points that you click on in the Drawing pane.

The cubic spline is not in the shapes menu; it is available by clicking **Right** on **[Create]** in the Drawing command menu. Place the cubic spline control points by clicking on the drawing window with the mouse ( see the section "Creating Shapes with the Graphic Editor", page 249).

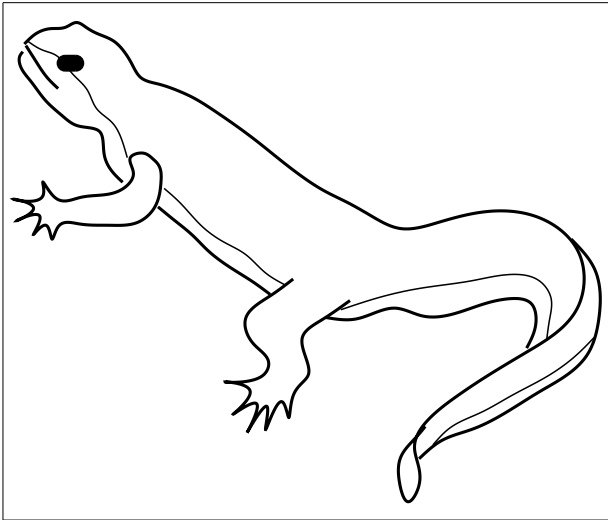
Here are the control points for a cubic spline (small x's indicate the points chosen so far):



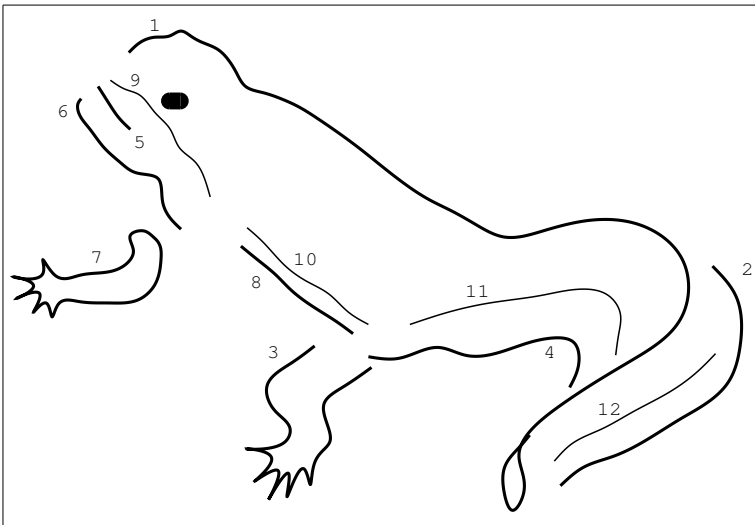
The cubic spline this creates looks like this:



You can create complex shapes using multiple cubic splines. For example:

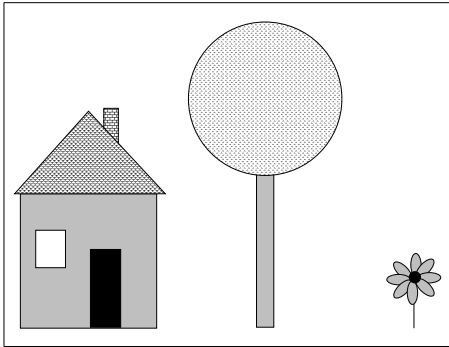


This picture uses 12 cubic splines and one small filled oval. Here is the newt decomposed into its 12 splines, numbered to indicate their order of creation.

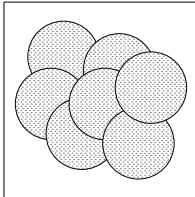


### 33.1.2. Combining Entities to Produce Complex Shapes

The most obvious combinations of simple shapes to produce more complex shapes are drawings done by small children:

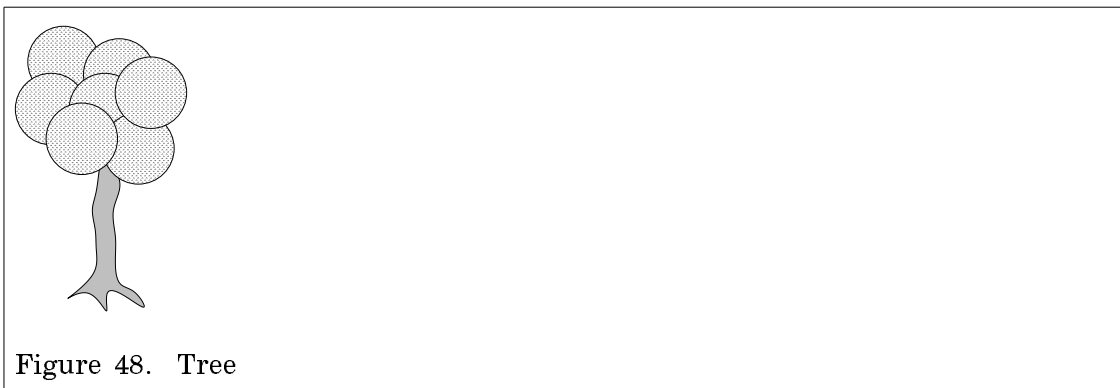


This is simple juxtaposition of shapes, with some overlapping. The top of the tree could be improved upon and made more lifelike if it were composed of a number of small circles, overlapped to simulate branch structure and introduce the suggestion of a third dimension:

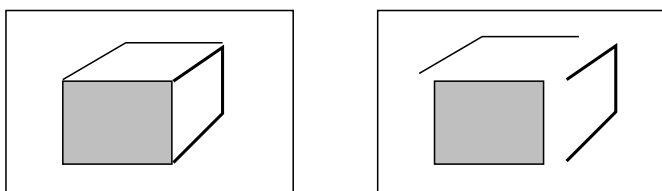


The overlapping circles can be *restacked* to give the tree a rounder shape. Restack Entities can be used to operate on all of the circles at once, or Bury and Raise can be used with individual circles.

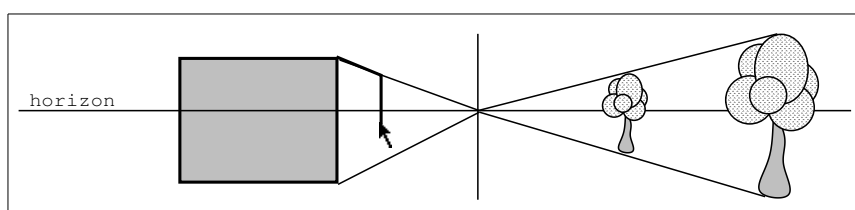
Now the tree looks more realistic, especially if its rectangular trunk is replaced by a closed cubic spline:



Shapes can be combined to produce perspective, as here for this cube, which is created from a rectangle and two unclosed polygons:



Temporary lines can be added to the drawing to serve as guides for drawing perspective.



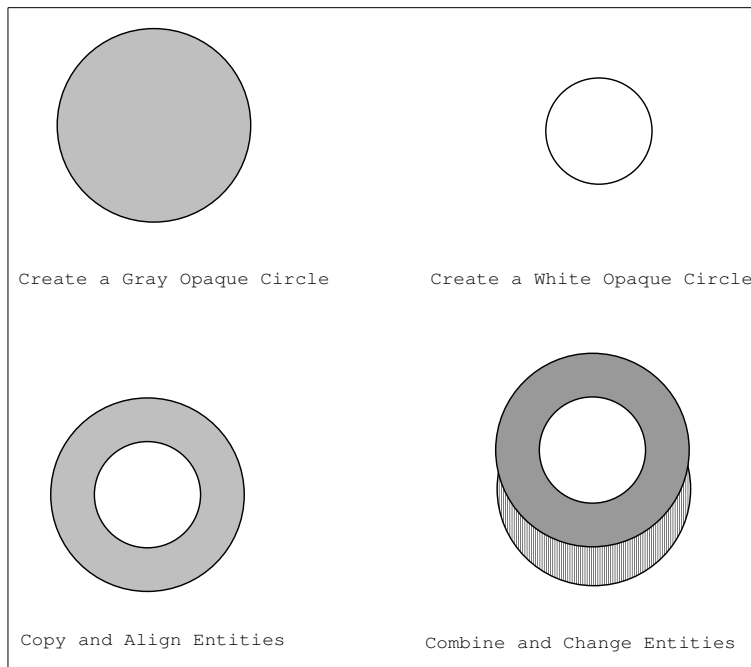
The Rename Entity command is useful here for naming these lines to things like Guide-1 and Guide-2 to make it easier to select them and delete them when you are finished.

### 33.1.3. Using Filling to Produce Shapes

You have three choices for the inside filling of an entity, Not Drawn, Opaque, and Non-Opaque. (Non-Opaque does not give predictable results with hardcopy.) If you choose Opaque, you can then select Black, White, Gray, or Patterned. The default filling is .25 Gray.

You similarly have the same three initial choices for the outline of an entity. For Opaque outlines you can then choose the line thickness and a dash pattern. The default outline is Opaque, not dashed, and thickness 1.

By combining filling and outline drawing, you can "cut out" parts of entities to create new shapes. For example:



1. Create a large gray circle by clicking Left on the Circle from the Shapes menu and by dragging (and releasing) the mouse on the Drawing window to define the location of the origin and the radius. The default fill attributes are:
  - Fill inside: Opaque
  - Fill pattern: Gray
  - Fill pattern gray level: .25
2. Create a smaller white circle using the method in Step 1. Click left on the entity to select it and then click Left on [Change] in the AVV menu to edit the default fill attributes. Change the fill pattern to white by clicking on [White] on the Fill pattern: line.
3. Copy the white circle onto the gray circle and align the entities on their centers. You can copy an entity by selecting it and dragging the mouse to the new location and releasing. Align multiple entities by selecting them and clicking Right on [Align] in the Drawing menu. You will be prompted for X and Y alignment parameters (in this case choose Center for both).
4. Combine these entities with other entities to produce complex drawings. You can select multiple entities by clicking  $\text{⌘}$ -Left on them in the Drawing pane. Then you can use other Drawing menu commands to operate on the selected entities together.



5. You can change the characteristics of individual graphic entities selecting them and using [Change]. In this example, the attributes of the bottom large rectangle have been changed as follows:

```
Fill inside: Opaque
Fill pattern: Patterned
Fill pattern pattern: Vert Lines
```

## 33.2. Grouping Entities

There are several reasons to group entities:

- To operate on several at the same time
- To more efficiently create repetitive patterns
- To keep the relationship between two or more entities constant

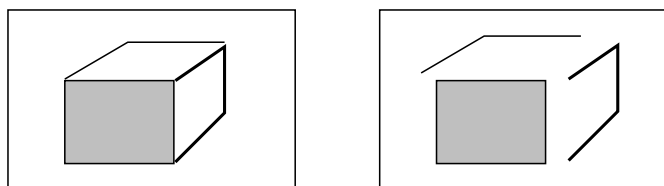
You can combine entities using any of the following methods:

- By clicking `⌘-Left` on the entities in the Drawing pane ( "Select Entity Graphic Editor Command").
- By defining a Region that entirely contains the entities ( "Select Region Graphic Editor Command").
- By grouping the selected entities ( "Group Entity Graphic Editor Command").

Entities that are combined using `⌘-Left` or are contained within a selected region are combined only while they are selected. When they are unselected, their association with each other is undone. This technique is useful for making the same changes to a number of similar entities.

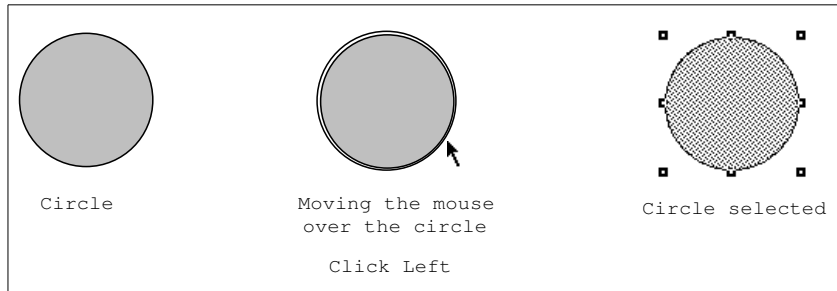
Entities that are combined using Group remain associated until they are explicitly Ungrouped. This is particularly useful for preserving carefully positioned entities that together make up a larger picture.

For example, this object is constructed of a rectangle and two polygons. For further drawing purposes, it might be useful to combine these entities for use as a cube.



### 33.3. Selecting Entities

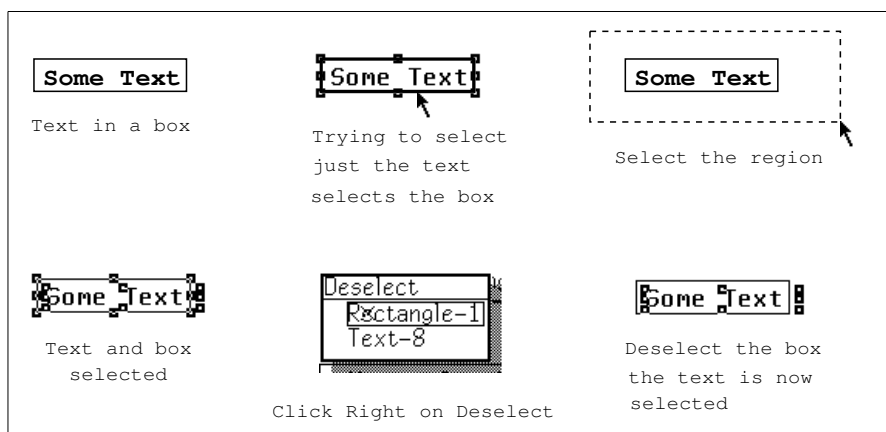
The easiest way to select an entity is to move the mouse over it and when it is highlighted, click Left (or  $\text{sh-Left}$  for multiple entities):



Sometimes, however, it is difficult to select an entity that is positioned very near or on top of another with the mouse alone. For example, if you have text on top of a rectangle (boxed text), or several overlapping entities, you sometimes cannot be precise enough with the mouse to select exactly the one you want.

In these cases, use the Select Region command and mark the region. Select Region only selects those entities that are completely within the region, so often you can pick up the entity you want by making sure it is completely within the region while its neighbors are not.

When entities overlap to such a degree that you cannot isolate one, select them all. Then you can click Right on [Deselect] and deselect the unwanted ones.



If you want to select one entity from a group, you must select the whole group and use Ungroup. Then you can select the individual entities.

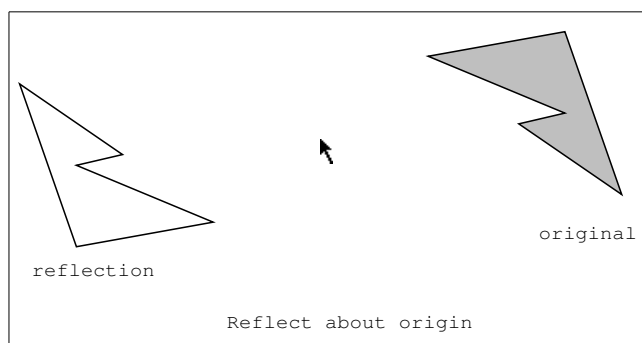
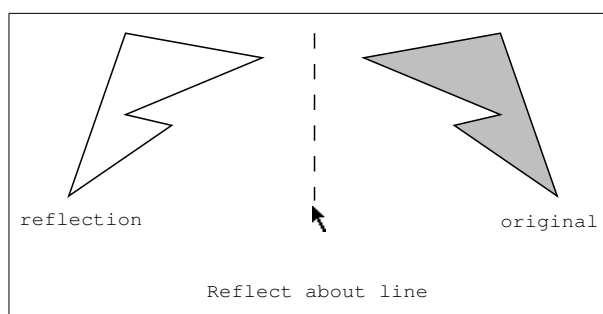
Another way to select a particular entity is to use the Select command and give it the name of an entity. All entities are automatically given unique names when they are created. Rename Entity allows you to give the entity a more easily remembered name.

Occasionally when you try some drawing operations, for example, drawing a line, you want to start the line on top of another entity or at the handle of another entity. When you position the mouse to start the line, you find that you cannot start line because you are selecting the other entity or operating on its handle. You can then click on the Line in the shapes menu or use Create Entity with an argument of Line to start the line drawing operation. Alternatively, you can draw a line of the proper length and orientation elsewhere on the drawing pane and move it into position.

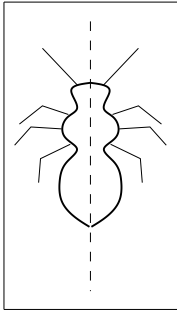
### 33.4. Transformations on Entities

Transform and Copy/Transform allow you to do two-dimensional transformations on one or more entities. Both commands perform the same transformations; the difference is that Transform performs the transformation on the selected entity or entities while Copy/Transform copies the selected entity or entities and performs the transformation on the copy.

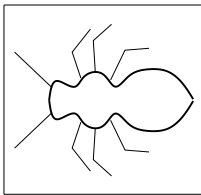
You can reflect an entity around a line or a point:



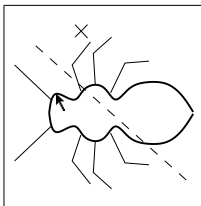
You can reflect an entity or entities with Copy/Transform to create symmetrical drawings. For example, half of this bug was drawn with cubic splines and lines. Then all the entities were selected and reflected using Copy/Transform around the dashed line:



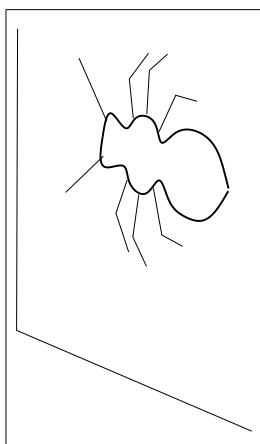
Drawing the bug vertically seemed the easiest way to properly visualize and create the shape. However, it takes up more room on the page that way, so once again all the entities in the drawing were selected and rotated 90 degrees using Transform:



You can use Shear and Stretch transformations to achieve some three dimensional effects. If we take the bug and use Shear with the control line as indicated by the dashed line, point to move as indicated by the cursor, and final location of the point (where the cursor ends up) as indicated by the X, like this:



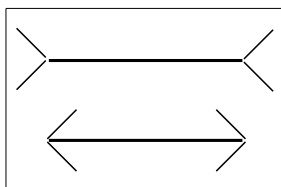
And then add a couple of lines, the bug appears to be crawling away across a wall.



### 33.5. Controlling the Size and Alignment of Entities

In artistic drawing the important thing to achieve is that an object look real, that the human eye is fooled into thinking that the sizing and alignments are accurate.

Exact measurement and precise alignment does not guarantee that things look the same. These two lines are the same length, being in fact copies of a single line:

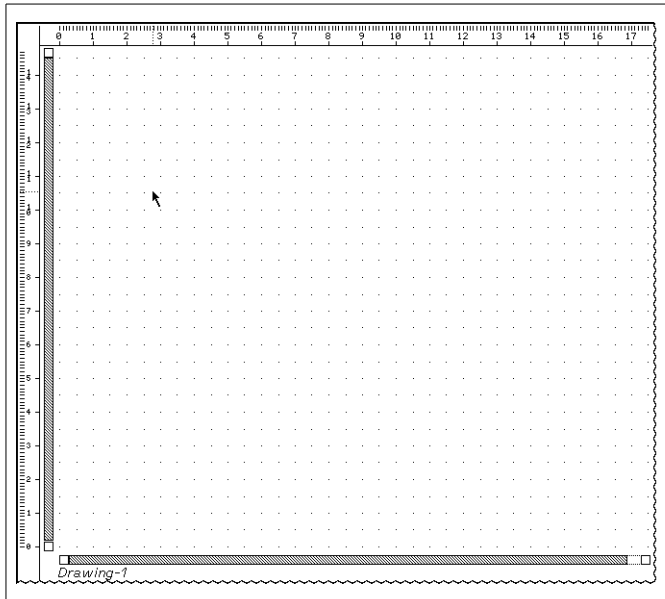


The simplest way to achieve uniformity among similar shapes in a drawing is to use [Copy] to create them. Create a single shape of the size you want, and then create additional ones by copying it with [Copy/Move] in the menu or by clicking Middle on the original entity.

The simplest way to position entities so that they line up is by using a grid. To display a grid, click on Display Grid: Yes in the Parameters menu. The default Grid Size is 1 centimeter. You can change the grid size by clicking on the value specifying the new grid spacing. (Note that smaller grid sizes take longer to display.)

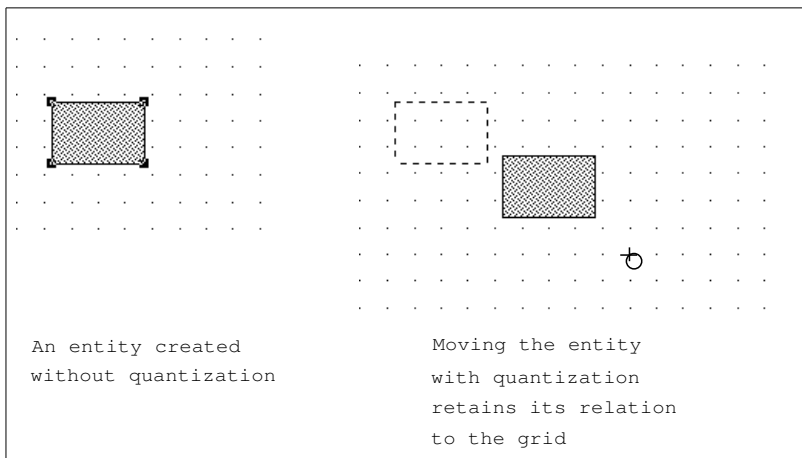
For more precise entity alignment you can display X- and Y-axis rulers by clicking on Display rulers: Yes in the Parameters menu. The default ruler units are centimeters. You can change the grid and/or ruler units using Edit Rulers.

With a grid size of 0.5 and rulers displayed, the drawing pane looks like this:



If you need more precision than aligning entities with the grid by hand, you can select **Quantize Mouse Position: Yes** in the Parameters menu. This automatically makes any mouse clicks align with the grid.

Note that when you use this option, only new entities will snap to grid points. Existing entities will not move relative to the grid.



If you want to align an existing entity with the grid, copy the entity to a register, delete it from the drawing, select Quantize mouse position: Yes, and retrieve the entity from the register.

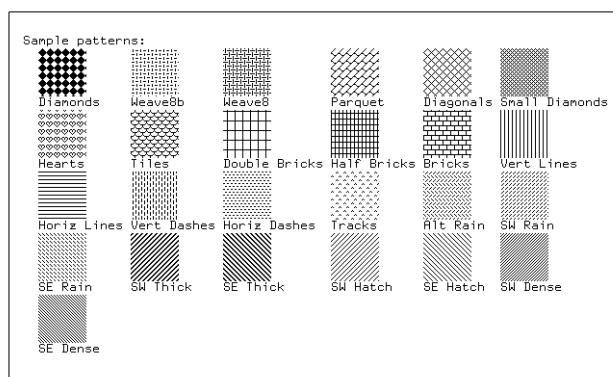
Sizing entities relative to each other can be accomplished by using the Scale option under [Transform] or [Copy/Transform]. See the Transform and Copy/Transform commands.

It is often easier to draw something large and then scale it down to the appropriate size. Alternatively, you can create the basic entity at an appropriate size and then use Zoom/Expand to enlarge it temporarily as you add detail. See the section "Changing the View of Drawings", page 265.

### 33.6. Filling a Shape Entity with a Pattern

You can specify the fill pattern or shading for Graphic Editor shape entities using [Change] command menu options. The current default filling for Graphic Editor shape entities shows in the Shapes menu at the bottom right of the screen (usually, a medium gray shading).

When you select the [Patterned] option, the menu of sample stipple patterns menu appears.



Note that if you have created a new stipple using the Stipple Editor, you can select it using this menu.

Use the following procedure to select a stipple pattern for filling selected Graphic Editor shape entities:

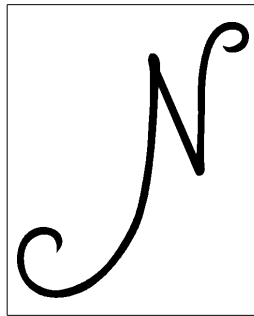
1. Click on [Change] in the Graphic Editor Entity Command menu.
2. Click on [Patterned] in the Change Options menu.
3. Click on the appropriate stipple pattern in the Sample Patterns menu (position the mouse so that the stipple pattern is highlighted by a rectangle before you


click on it). Note that you may need to scroll the Change options menu to see samples of all of the available stipple patterns.

4. Click on END to exit the menu and to apply the options you selected.

### 33.7. Example of Using the Graphic Editor

This example shows how to use the Graphic and Bitmap Editors to make the following drawing.



Select the Graphic Editor activity by clicking on  in the Symbolics Concordia icon menu at the top-right of the screen, or by pressing SELECT G.

When you create complex drawings, we recommend that you use the following Graphic Editor display options. (Set them by clicking on them in the Parameters menu to the right of the Drawing pane).

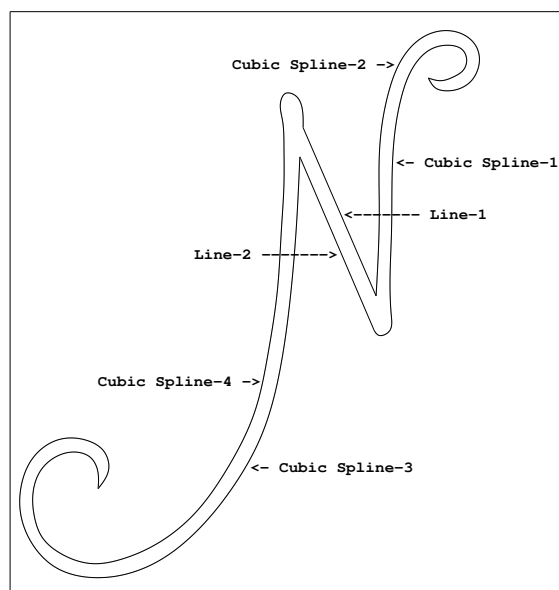
Display grid: **Yes**  
 Show rulers: **Yes**

Select a new drawing by clicking on [Select Drawing] and specifying the drawing name. You can also [Delete All] to clear the currently active drawing. Use this drawing name in the Symbolics Concordia  $m-x$  Create Picture command to include the drawing in a document.

[Select Drawing] N

This illustration shows the graphic entities that make up the drawing (four cubic splines and two line entities).





Control points define the shape of certain Graphic Editor entities (such as cubic splines). A bounding box is the smallest possible rectangle that contains the entity.

You can display the control points of a shape entity by selecting [Control points] or [Control if selected] display options in the Parameters menu. You can display the bounding box by selecting [Bounding box] or [Box if selected] display parameters options. The "if selected" options display only for entities that are selected in the drawing pane.

Handles: Control points	Control if selected
Bounding box	Box if selected

To create a cubic spline entity click Right on [Create] and then click on [Cubic Spline] or type the Graphic Editor command

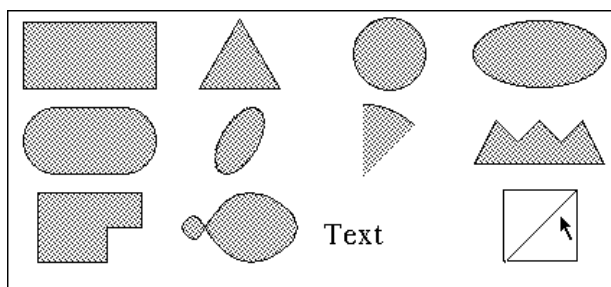
Create Entity Cubic Spline

Click Left on successive control points to define the curve (click  $\text{⇧-Left}$  to finish).

```
click Right on [Create]
[Cubic Spline]
{click on control points to define the curve}
```

Note that you can edit the shape of most graphic entities by clicking on a control point and dragging it to a new position. Using control points to define cubic splines and Be'zier curves is a particularly effective approach, because it allows you to make small changes to the shape of the curve easily.

To create a line entity click on the line (/) in the shapes menu at the bottom right of the screen.



Position the mouse in the drawing pane on the starting point of the line and click Left (and hold). Drag the mouse and then release to define the line.

You can add annotation to the drawing by selecting [Text] in the shapes menu (bottom right of the screen). You are prompted for the text to insert and for the location in the drawing.

To correct the alignment or to perform any other transformation on an entity (or group of entities), click Right on it when it is highlighted in the drawing window. This displays the possible operations you can perform on the entity. Then you can click on the operation you want to perform (such as Move, Copy, or Change).

Note that for many Graphic Editor commands you are prompted for additional keyboard information or mouse gestures (such as the text string to insert, or a control point for a cubic spline entity). Look in the prompt window for Graphic Editor keyboard prompts. The interpretation of current mouse gestures appears in the mouse documentation line at the bottom of the screen.

Before you can add a drawing to a Symbolics Concordia document, you should first associate the drawing with a drawing file. You can do this with the "Set Drawing File Graphic Editor Command" or with the "Write File Graphic Editor Command".

```
Graphic editor command: Set Drawing File
                        {drawing-name} {file-pathname}
```

Uneditable versions of the drawings in Concordia documents are automatically saved when you save the document. You do not need to save drawings in a separate drawing file unless they are particularly difficult to reproduce and you expect to edit them the future. Saving drawing files takes a lot of disk space.

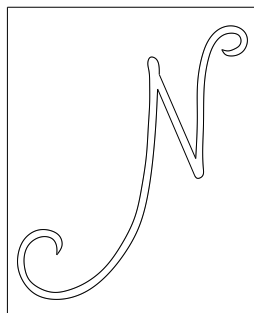
To exit from the Graphic Editor select [Done]. This returns you to the previous activity.

```
[Done]
```

Use the `m-x` "Create Picture" command in your Concordia document to insert the drawing into your document. This creates an icon in your document that represents the drawing.

```
☞ Picture "N" from file SAP:>sys>graphic-editor>documentation.pic.newest
```

The picture icon is interpreted when the document is formatted to display your drawing.



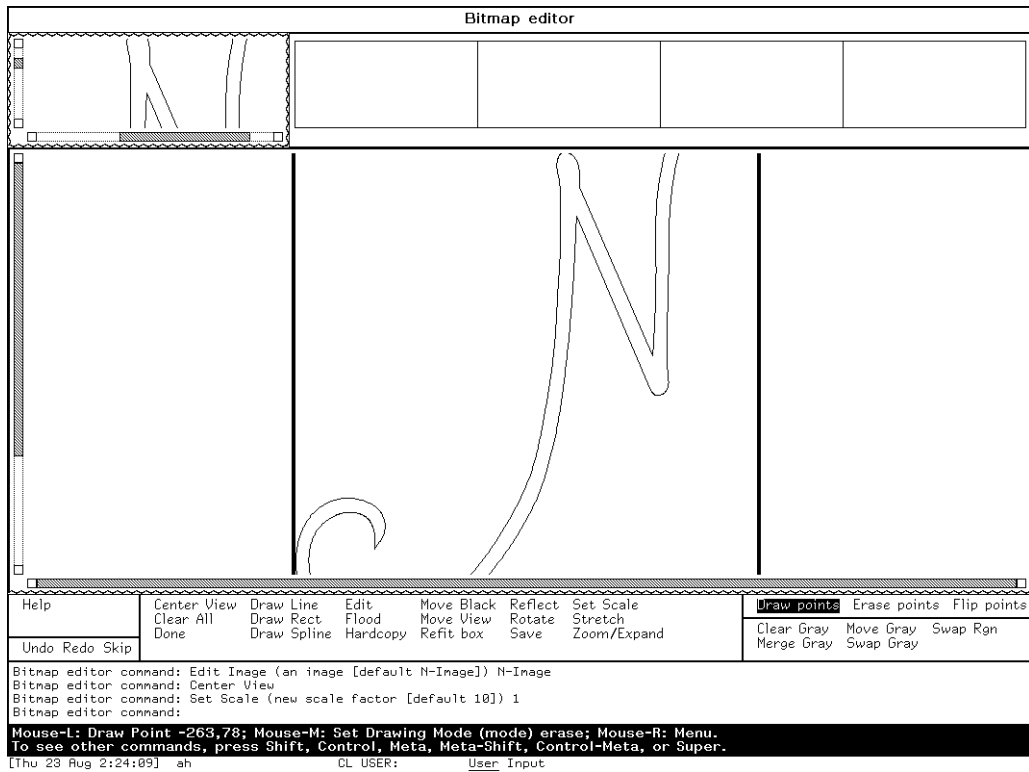
For more information on using drawings as figures in Symbolics Concordia documents, see the section "Including a Graphic Editor Drawing in a Document", page 270.

Note that you cannot use the Graphic Editor to fill arbitrary closed spaces that are bounded by cubic splines or other line entities, because the editor does not understand what "inside" means between multiple entities. However, you can fill these spaces with the "Flood Region Bitmap Editor Command".

Use the "Copy Drawing to Image Graphic Editor Command" to make a bitmap image of the drawing. The default name of the image is *drawing-name-Image*.

Copy Drawing to Image *image-name scale*

Then, select the Bitmap Editor activity ( using `SELECT }` ) so that you can edit the bitmap image. Use the "Edit Image Bitmap Editor Command" to bring the image into the drawing pane for editing. Note that the image you just created in the Graphic Editor is the default.



Note that you may need to use [Center View] and [Set Scale] to reposition the image in the Bitmap Editor drawing pane.

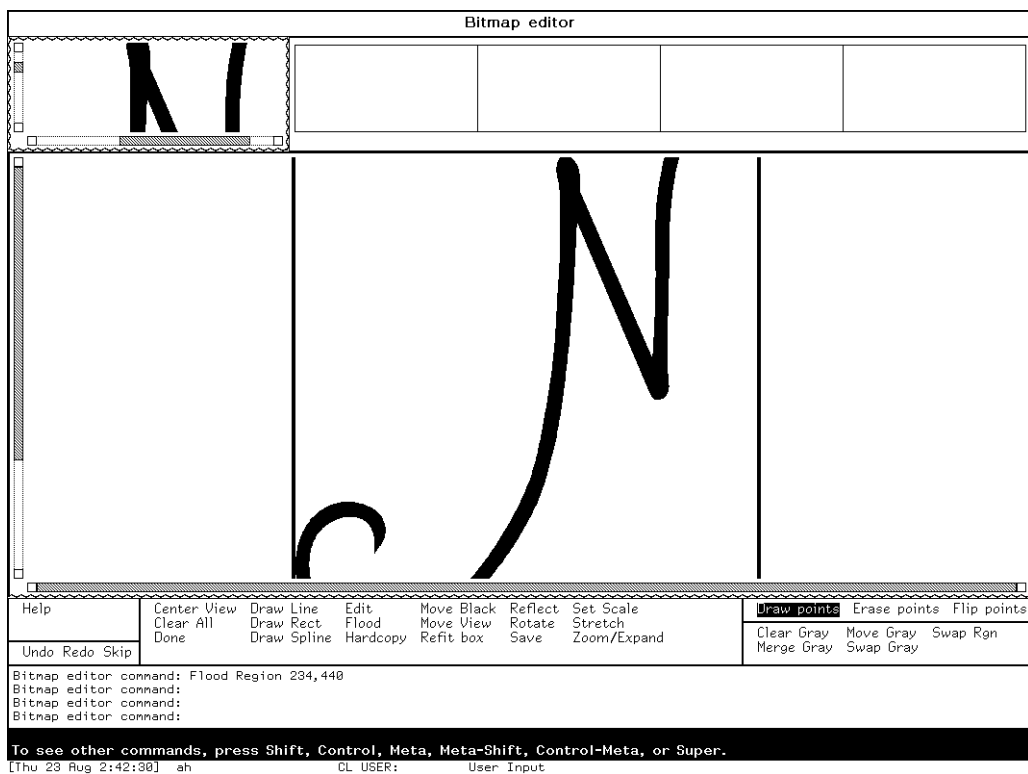
You can then use [Flood Region] to fill the bounded region (click on a seed point inside the region you want to fill and press RETURN). Be sure that the region is closed (that is, that there are no gaps in the boundaries, particularly where lines join). You now have an image that is just like the original drawing, but filled.

When you click on [Done], the Graphic Editor activity resumes.

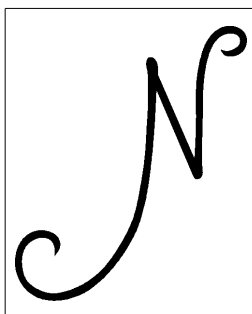
```

SELECT }
[Center View]
[Set Scale] 1
Edit Image image-name
Flood Region {click on a seed point and press RETURN}
Done

```



Use the "Add Image Graphic Editor Command" to put the image into a Graphic Editor drawing (click to anchor it in the drawing frame).



For more information on using drawings as figures in Symbolics Concordia documents, see the section "Including a Graphic Editor Drawing in a Document", page 270.



## 34. Dictionary of Graphic Editor Commands

**Add Image** Inserts a bitmap image picked up from the screen or from the bitmap editor into the current picture. It prompts for a named image, with the default being the current image.

### *Example*

This example captures a screen image using the Bitmap Editor (FUNCTION Ø Q) and transfers it to the Graphic Editor (SELECT G). You can then use the Graphic Editor to modify the image and to make pictures to include in Symbolics Concordia documents.

```

FUNCTION Ø Q
  Destination: [Named Image]
  Name: {picture-identifier}
SELECT G
  Graphic Editor command: Add Image {picture-identifier}
SELECT W
  m-X Create Picture
    Picture type {Graphic-editor}
    Name of drawing {picture-identifier}

```

**Align Entity** [Align] Aligns one or more entities to each other and optionally to a specified screen grid point. You can select several entities to be aligned by positioning the mouse over each entity and pressing `sh-Left`.

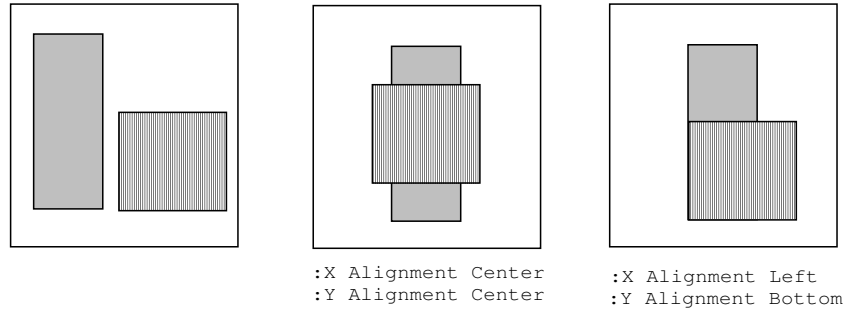
This command accepts the following keywords:

:X Alignment (left, right, center, none). The default is left.

:Y Alignment (top, bottom, center, none). The default is none.

:Align To Grid (yes, no) Specify the grid point in integer screen coordinates. The default is yes.

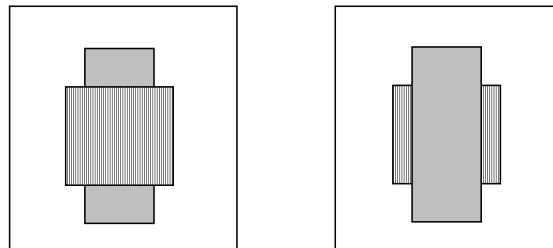
### *Example*

**Bury Entity**

[Bury] Moves the chosen entities under all others it overlaps (the opposite of the "Raise Entity Graphic Editor Command").

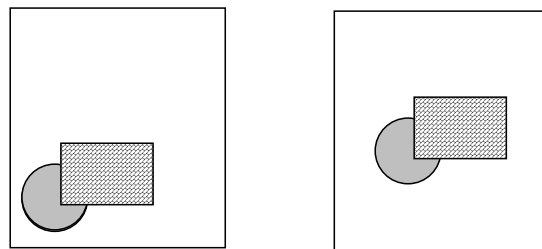
*Example*

This example buries the striped rectangle.

**Center View**

[Center View] Adjusts the translation of the drawing or image display to align the center of the drawing or image with the center of the drawing pane.

This adjustment is for display purposes only and does not affect the actual size of entities in the drawing or image.

*Example***Change Entity**

[Change] (You can also use `c-m-Right` for all except text entities and images). Changes the display characteristics of one or



more entities. For graphic entities you can change display characteristics such as, outline thickness, line dashing, and filling patterns, gray density, and presentation object. For text entities, you can change font, size, orientation, and fill.

You can also change the presentation type and object of any graphic entity.

The Presentation Type determines where and when a graphic entity is mouse-sensitive and what kinds of presentation objects are valid. The Presentation Object names a specific presentation object and controls what happens when the entity is clicked on.

See the section "Controlling Mouse-Sensitivity in Graphic Editor Drawings", page 245.

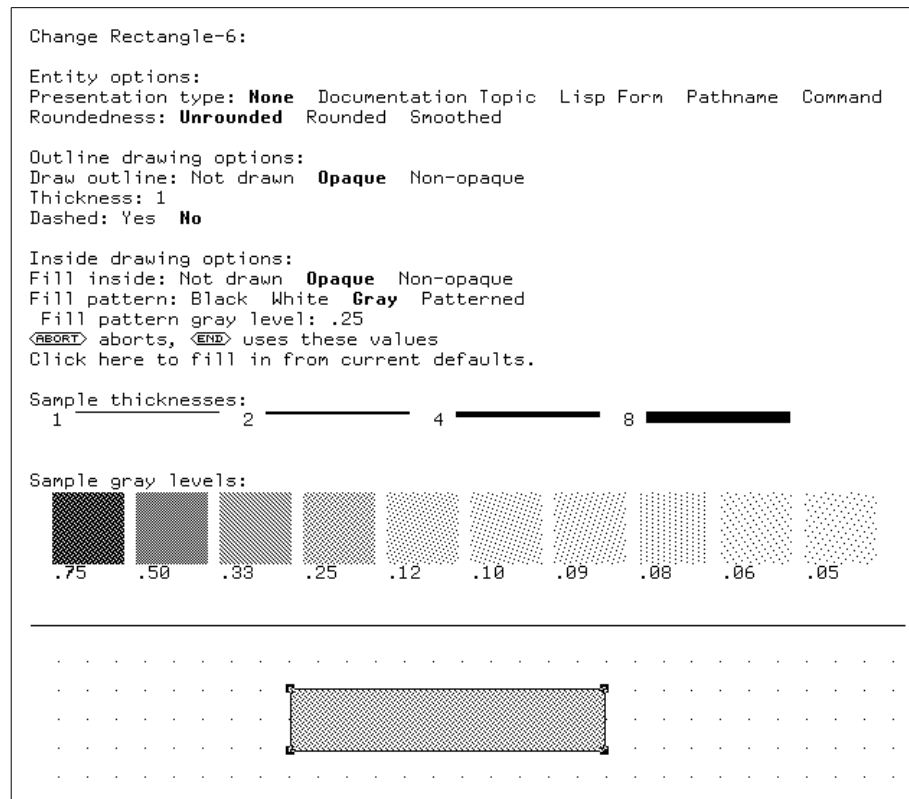
To see the display characteristics for an entity select it and click Right on [Change] in the Drawing command menu.

The normal AVV display of options is followed by sets of samples for those item types that are visible, for instance, sample line thicknesses and gray levels. These samples are mouse sensitive and will alter a visible query of the given type.

You can create your own custom patterns for filling shapes using the "Define New Pattern Graphic Editor Command".

### *Example*

This screen image shows the menu of display attributes for the selected rectangle entity (shown at the bottom of the drawing window).



**Change Text String** Changes the text of a selected text entity. You can select the text entity by clicking on it in the drawing window or by naming it in the Change Entity command (in the command echo pane at the bottom of the screen).

You can use Change Entity if you want to change the display characteristics of text entities (such as font, size, orientation, or fill).

**Clear Undo History** Clears everything out of the undo history. The operations in the undo history then become accessible to the garbage collector.

**Clear Register** Clears out the specified register (registers are numbered sequentially with the leftmost register named Register-1). The usual way to execute this command is to position the mouse over the register so that its borders are darker. Click Right for the overview menu and then click Left on [Clear Register].

For more information, see "Store into Register Graphic Editor Command", and "Retrieve Register Graphic Editor Command".

**Command Help** See "Help Graphic Editor Command"

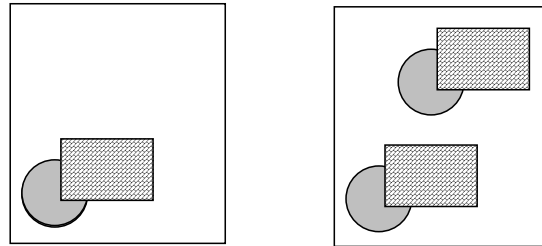
**Copy and Move Entity**

[Copy/Move] Makes a copy of the specified entities and places

it where you click the mouse. Also accessible by clicking Middle on an entity and holding. See the section "Using the Mouse in the Graphic Editor", page 238.

### Example

In this example both entities were selected and Move/Copied together by clicking  $\text{sh-Left}$  and then clicking on [Move/Copy] in the Drawing command menu (clicking-Left again anchors the copy at the current drawing location).



### Copy and Transform Entity

[Copy/Transform] Makes a copy of the specified entities and applies the specified transformation to them. You can select the entities in the drawing pane, click Right for the entity operations AVV menu, and click Left on [Copy/Transform], or you can select the entities and click Left on [Copy/Transform] in the drawing command menu.

For more information, see "Transform Entity Graphic Editor Command".

### Copy Drawing

Makes a copy of a drawing. You can use this command to quickly and easily make a series of drawings that differ only slightly. You are prompted for the name of the drawing to copy, and for the name of the new drawing.

Copy Drawing *{original-drawing-name} {new-drawing-name}*

The new drawing will appear in the Graphic Editor window. You can then edit the drawing and save it ("Write File Graphic Editor Command").

### Copy Drawing to Image

Makes a bitmap image of the current Graphic Editor drawing. The first argument is the name of the new image. The second argument is a scale factor (the default of 1 indicates full size).

Copy Drawing to Image my-new-image 2.5

Note that the new image will be the the default image for the "Edit Image Bitmap Editor Command".

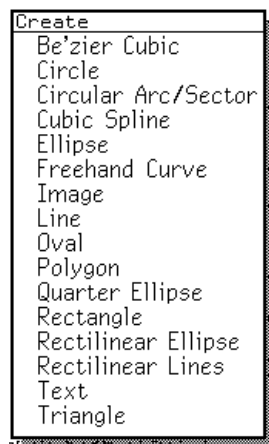
**Copy Entity** [Copy] Makes copies of the selected entities. Use Select Region or `sh-Left` to identify multiple entities to be copied together. When you hold the mouse button down, the selected entities will be Copy/Moved to the drawing location where the mouse is released. If you click on [Copy] in the drawing command menu the selected entities will be copied and moved to a location close to the original entities.

**Create Entity** [Create] Creates a new graphic entity. Note that any defaults that were specified using the "Edit Defaults Graphic Editor Command" ( [Defaults] in the command menu) are applied to new entities.

Use any of the following methods to create a new graphic entity:

- Click Left on one of the shapes in the shapes menu at the lower right of the Graphic Editor screen.
- Click Right on [Create] in the drawing command menu (or enter Create Entity command via the keyboard) and then click Right for the AVV menu of shapes. Position the mouse on the desired shape and click to select it.
- To create a line entity you can click Middle in a blank area of the drawing.

This illustration shows the Create Shapes AVV menu. Click on the entity type to select it.



When you create an entity, its identifier is reported in the command echo pane. For example,

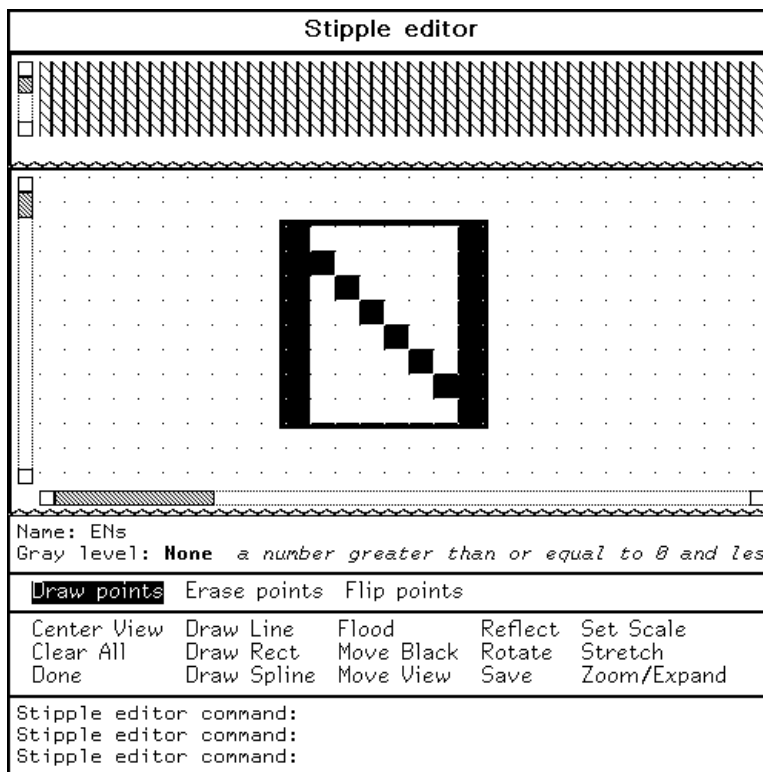
```
Graphic Editor command: Create Entity Rectangle
Rectangle-1 created.
```

You can use this identifier to refer to this entity in other Graphic Editor commands, or you can rename it to something more descriptive ("Rename Entity Graphic Editor Command").

After you select an entity type to be created you may be prompted for additional information to define the exact shape, orientation, and location of the new entity. See the section "Creating Shapes with the Graphic Editor", page 249.

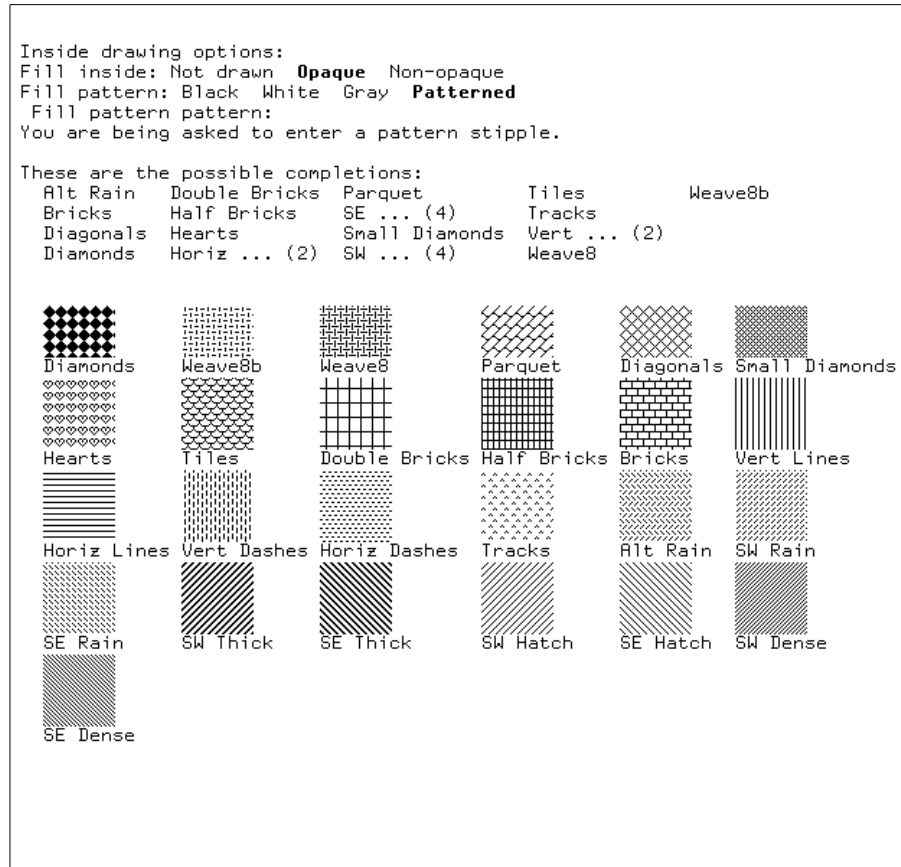
### Define New Pattern

Invokes the Stipple Editor for defining a new custom fill pattern. Fill patterns are selectable from the Change Entity and Edit Defaults command menus when you select the **Patterned** option.



When you click on [Save] and then [Done] in the Stipple Editor command menu, the pattern you define will be added to the set of patterns available to the [Change] and [Defaults] Graphic Editor commands.

The following figure shows a portion of the [Change] menu which displays the default fill patterns.



The Stipple Editor is similar to the Bitmap Editor. See the section "Using the Bitmap Editor", page 317.

- |                 |   |
|-----------------|---|
| Delete All      | [Delete All] Clears the contents of the current drawing.  |
| Delete Entity   | [Delete] Deletes one or more entities from the drawing. It operates on the selected entities, or, if no entity is selected, it prompts for an entity.   |
| Deselect Entity | [Deselect] Unselects a set of entities. Undoes the action of select. The default is the whole selected set (you can also click Left in a blank area of the drawing to deselect all of the selected entities).<br><br>You can also click Right on [Deselect] for a menu of entities to deselect. |
| Done            | [Done] Exits the program, returning you to your previous activity.  |

For example, If you were editing a picture (that is, you entered the Graphic Editor by clicking  $m$ -Left on a picture in a Symbolics Concordia document) and then you click on [Done] in the

Graphic Editor, control returns to the Concordia editor (the edited version of the picture is automatically recompiled and inserted in the record).

### Edit Defaults

[Defaults] Changes the default drawing parameters to be used for new entities. These defaults will all apply to all newly created drawing entities.

To change the appearance or presentation parameters of existing entities use [Change].

[Defaults] pops up a menu, which looks like this:

```

Polygon roundedness: Unrounded Rounded Smoothed
Text character style: NIL.NIL.NIL
Arrowhead(s): Start End
Presentation type: None Documentation Topic Lisp Form Pathname Command

Outline drawing defaults:
Draw outline: Not drawn Opaque Non-opaque
Thickness: 1
Dashed: Yes No

Inside drawing defaults:
Fill inside: Not drawn Opaque Non-opaque
Fill pattern: Black White Gray Patterned
Fill pattern gray level: .25

Other command defaults:
Zoom factor: 4
Hardcopy if too large: Clip Scale Multiple-Pages
Hardcopy orientation: Landscape Portrait
Rotation angle: 90

Other drawing defaults:
Draw faster rather than more accurately: Yes No
<REORT> aborts, <END> uses these values

Sample thicknesses:
1 _____ 2 _____ 4 _____ 8 _____

Sample gray levels:
[ .75 [ .50 [ .33 [ .25 [ .12 [ .10 [ .09 [ .08 [ .06 [ .05

```

The sample shapes in the shapes menu at the bottom right of the window are displayed using the defaults currently in force.

See also "Edit Interaction Style Graphic Editor Command" and "Edit Ruler Graphic Editor Command" for more details on customizing the Graphic Editor.

**Polygon roundness** How the corners of polygon entities are drawn. The options are Unrounded, Rounded, or Smoothed.

**Text character style**

The character style of text entities. Character styles are in the form `family.face.size` (for example, `FIX.ROMAN.NORMAL`). For more information, see the section "What is a Character Style?" in *Genera User's Guide*.

**Arrowhead(s)**

How to draw arrows. You can put arrowheads on either the Start, End, or both ends of lines. When you select either of these arrowhead options, you can also choose to specify the dimensions of the arrowheads.

**Presentation Type**

Determines where and when a graphic entity is mouse-sensitive and what kinds of presentation objects are valid. For more information, see the section "Presentation Types for Mouse Sensitive Text and Drawings", page 247.

**Presentation Object**

Names a specific presentation object and controls what happens when the entity is clicked on.

**Outline Drawing Defaults****Draw outline**

Outlines for shapes entities can be Not drawn, Opaque, or Non-opaque.

**Thickness**

The line thickness of drawn shapes outlines.

**Dashed**

You can choose to draw shapes outlines with dashed or undashed lines.

**Line pattern**

You can choose Black, White, Gray, or Patterned. Note that these options only appear in the AVV menu when you change the outline line Thickness value.

**Line end shape**

You can choose Butt, Square, or Round. Note that these options only appear in the AVV menu when you change the outline line Thickness value.

**Line joint shape**

You can choose Mitre, Bevel, Round, or None. Note that these options only appear in the AVV menu when you change the outline line Thickness value.

**Inside Drawing Defaults**



- Fill inside** You can choose how to fill shape entities (Not drawn, Opaque, Non-opaque). If you choose Opaque, shapes entities will hide any entities "behind" them. Non-opaque entities are "transparent".
- Fill pattern** You can fill shapes with color (Black, White, or shades of Gray), or with a Pattern. A menu of sample gray patterns for you to choose from appears at the bottom of the Defaults menu. Gray .25 is the default in this menu. If you choose Patterned, a sample menu of patterns appears (you can click on one of the sample patterns to select it.)
- Note that the shapes menu shows the current inside drawing defaults.

### Other Command Defaults

- Zoom factor** The default scale factor for the "Zoom by Factor Graphic Editor Command".
- Hardcopy if too large** How to print drawings that are too large to fit on a page. You can select Clip, Scale, or Multi-pages.
- Hardcopy orientation** You can choose to print drawings Landscape, or Portrait.
- Rotation angle** This is the default  $n$  for the [Rotate  $n$  Degrees] Interactive Transform Type option of the "Transform Entity Graphic Editor Command".

### Other Drawing Defaults

- Draw faster rather than more accurately** You can choose to speed up the display of drawings (at the expense of accuracy).

## Edit Image

[Edit] Places a bitmap image into the Bitmap Editor for editing. It prompts for a named image, with the default being the current image. You can also initiate this command by clicking `c-n-Right` on an image entity in the drawing pane.

You can select an image by typing its name or clicking on it with the mouse.

Note that this command accepts only images. If you want to create an image from a drawing use the "Copy Drawing to Image Graphic Editor Command".

The Bitmap Editor is selected and the image is placed in it for editing. If the real bitmap is cached in core (as a named image or the result of editing a named image picked up from the screen), that is used. If the image is one that has been stored in a Graphic Editor picture, it has become a single entity rather than a bitmap. A bitmap is generated from this entity for you to edit. Note that there might be some loss of resolution in the translation.

When you are finished editing the image, click on [Save] to cache the new image, and then on [Done] to return it to the Graphic Editor.

### Edit Interaction Style

Changes the default user interface to some drawing operations. When you enter this command, this menu of options is displayed:

```

Erase before moving: Don't erase  Erase  Partially-Erase
Moving feedback:  Image  Outline
Select origin for rotate, scale, etc.: Choose  Center  Corner
Text entry mode:  AVV menu  Line of text
Circular arc input: Center and two radii  Two endpoints and bow
Default highlighting: Outline  Bounding-Box
Some command options defaults are sticky: Yes  No
<ABORT> aborts, <END> uses these values

```

#### Erase before moving

Whether an object that is being moved should be erased. The choices are Don't Erase, Erase, and Partially Erase. The default is Erase.

#### Moving feedback

What the visual feedback during the moving operation should be. The choices are Image and Outline. The default is Image.

#### Select origin

What the center of rotation should be. The choices are Choose, Center, and Corner. The default is Choose.

#### Text entry mode

How text is created. The choices are from an AVV menu or line of text from the minibuffer. The default is from the minibuffer.

#### Circular arc input

How arcs are created. The choices are [Center and two radii] and [Two endpoints and bow]. The default is Two endpoints and bow. For more information, see "The Circular Arc/Sector Graphic Editor Shape".

**Default highlighting**

How entities are highlighted when the mouse is over them. The choices are Outline and Bounding Box. The default is Outline.

**Sticky defaulting**

Whether arguments for some command options should have sticky defaults, that is, once you specify an option it becomes the default for the next use of that command. The default is No.

To change the appearance of existing entities, use Change. To change the appearance of entities you subsequently create, use Defaults.

**Edit Ruler**

Allows changing how the ruler is displayed. When you enter this command, this menu of options is displayed:

```
Scale unit: Centimeter Inch Pixel
Display rulers: Yes No
Units per major division: 1
Units per minor division: 1/10
Grid size: 1
Display hardcopy box: Yes No
<REORT> aborts, <END> uses these values
```

**Scale unit**

The units used, centimeters, inches, or pixels.

**Display rulers**

Specify whether the rulers are displayed.

**Units per major division**

Ruler scale increment for labeled units (for example 1 to label each whole unit on the ruler).

**Units per minor division**

Ruler scale increment for ticks between units (for example .1 for 10 tick marks between each unit on the ruler).

**Grid size**

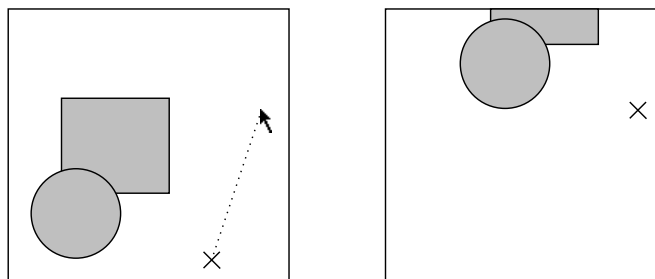
The grid size (when the grid is displayed).

**Display hardcopy box**

Whether the hardcopy box is displayed. The hardcopy box shows the page boundaries on the drawing pane.

- Fit View** [Fit View] Adjusts the scale and translation of the drawing display to exactly fit into the drawing window.
- This adjustment is for display purposes only and does not affect the actual size of entities in the drawing or image.
- Group Entity** [Group] Groups a set of entities together. You can select multiple entities using `⇧-Left` on each entity to be included in the group. Once grouped, they act as a single entity for all commands. Use `Ungroup` to separate a group into individual entities.
- Hardcopy Drawing** [Hardcopy] Hardcopies the current drawing. You can optionally specify the printer.
- Hardcopy File** Hardcopies the drawings in the file you specify. If the file is not loaded, it is read in first.
- Help** [Help] The Help command prints out help on other commands, using the standard documentation database and formatter.
- After clicking on the Help command, you can enter a topic, such as a command name, from the keyboard, or select a menu item from the editor's command menus.
- Kill Drawing** [Kill Drawing] Kills a drawing buffer. If you kill the current drawing, the previously selected drawing is selected. If there is no previously selected drawing (this is, you kill the last drawing), a new drawing is created with a unique name.
- The drawing is not lost forever. The command is undoable, as long as you have not used `Clear Undo History`.
- Move Entity** [Move] Moves the selected entities with the mouse. Also accessible by clicking-Left on an entity and dragging to the new position (release the mouse button to anchor the entities at the new position). Note that this command does not make a copy of the selected entities. It moves the entities to a new location in the drawing. Use [Copy/Move] if you want to duplicate the selected entities in a new location in the drawing.
- See the section "Using the Mouse in the Graphic Editor", page 238.
- Move Entity Handle** Drags one handle of an entity, reshaping the entity.
- This command is available by clicking Left and holding on a handle of an entity. It cannot be typed from the keyboard.
- Move View** [Move View] Adjusts the view of the drawing or image. The drawing is adjusted so that the first point you click on in the

drawing pane is shifted to the second point in the pane that you click on. For example:



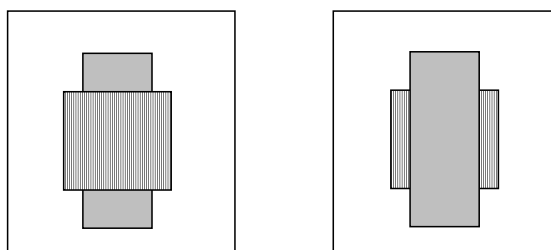
The location of the first click is indicated by the X, the second by the cursor location. The drawing is scrolled up and to the right, as shown (the X is left in the picture for reference, on the screen it disappears).

### Raise Entity

[Raise] Moves the chosen entities to the front of the display (the opposite of the "Bury Entity Graphic Editor Command").

#### *Example*

This example raises the solid rectangle.



### Read File

[Read File] Reads in a file full of drawing buffers, then selects one of the drawings in the file. When you click Right on [Select Drawing] you will see a list of the drawings in all of the files you have read.

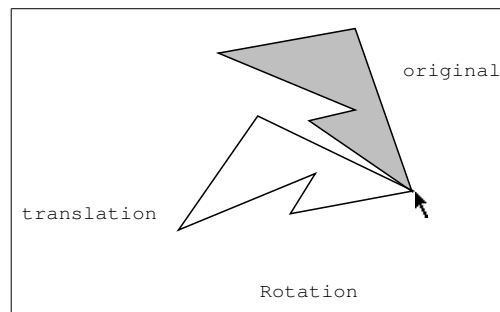
### Redo

[Redo] Redoes the last command that was undone via the Undo command.

It is possible to have more than one command that got done after the command that you have undone back to. In this case, a menu is popped up. Newer commands are at the bottom.

Refresh	[Refresh] Redraws the drawing or image display.
Rename Drawing	[Rename Drawing] Changes the name of the current drawing.
Rename Entity	Changes the name of a selected entity. Graphic Editor entities are assigned unique names which consist of the shape name followed by "-" and an increasing number (for example Rectangle-1). For a display of the entity names in the active drawing, click Right on [Select]. The name of a selected entity is displayed in the command echo window at the bottom of the screen.
Reset View	[Reset View] Resets the drawing display to normal scale and translation. This resets the effects of scrolling around or of using Change View, Fit View, or Move View.
Restack Entities	Changes the order of entities overlapping the chosen one. Change an entity stacking location by clicking on its name in the entity menu.
Retrieve Register	Retrieves the contents of the register. The contents are put down where you click with the mouse.
Rotate Entity	Rotates the selected entities about a particular point. Click Left on the point to be the center of rotation and hold down the button as you move the mouse around that center. Release when you have the desired rotation.

### *Example*



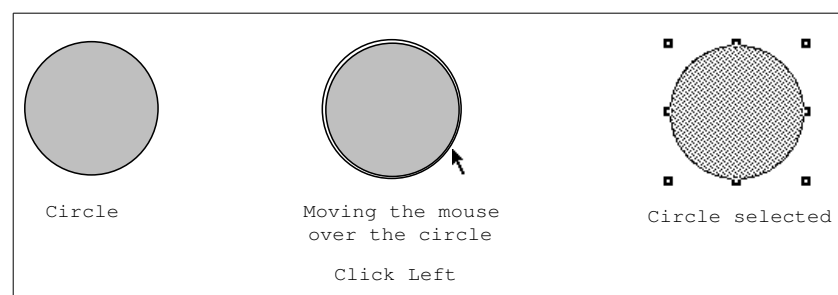
Save Drawing	Saves drawing(s) to the specified file. You can specify a particular drawing or All. The default is the active drawing. Note that only drawings that are saved can be edited.
--------------	--

- Select All** [Select All] Selects all entities in the current drawing.
- Select Drawing** [Select Drawing] Selects a drawing buffer. You can see the drawings already in Graphic Editor buffers by clicking-Right on [Select Drawing] in the drawing command pane.
- Select Entity** Selects an entity or a set of entities. You are prompted for an entity. This command is also accessible by clicking-Left on a displayed entity. Clicking  $\text{sh}$ -Left on an entity adds it to the set of selected entities.

The selected entity or set of entities is normally highlighted by little boxes, called *handles*, on them ("Edit Defaults Graphic Editor Command").

This set of the selected entities is the default for most commands that act on entities. Selecting entities to move them around is a common operation, so there is a single command to do that, Select Entity and Move Selected.

The following illustration shows how to select a circle.



### Select Entity and Move

Selects an entity or a set of entities and picks them up for moving. You can move multiple entities by selecting them clicking  $\text{sh}$ -Left and then by clicking on [Move] in the command menu. Click again when the entities are positioned where you want them in the drawing pane.

### Select Region

Selects entities entirely contained within (that is, not overlapping) a specified rectangle. To define the selected region click on the upper left of the rectangle and drag the mouse down to the lower right and release.

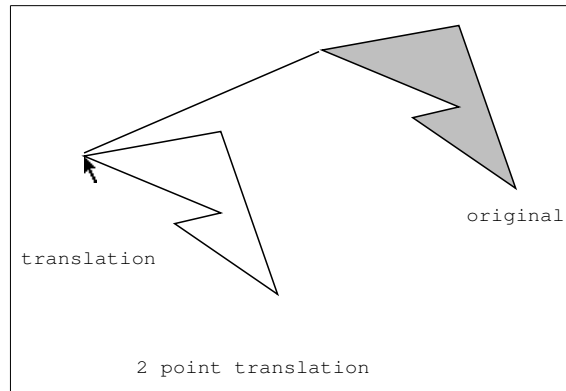
This command is also accessible by clicking Left somewhere in the drawing where nothing else is displayed. In that case, you hold the left button down while sweeping out the rectangle.

- Set Drawing File** Associates a drawing with a file. If the file already exists, you are asked if you want to read the file in ("Read File Graphic Editor Command").
- Note that this command does not save the drawing.
- Set Drawing File is particularly useful when you are creating multiple drawings. When you have completed editing a drawing, use Set Drawing File to associate it with a file. When you have completed editing all of your drawings, use the "Write File Graphic Editor Command" to save the drawings associated with the file. This minimizes saving of multiple copies of the drawing file
- Show Documentation** Prompts for a record and displays its installed documentation in a typeout window. Also available by clicking Left on a record in the Previewer. Use the **:destination** keyword option to send output to the printer of your choice. Note that Show Documentation prints text and illustrations but does not print section numbers, page headings, and so on.
- Show File Drawings** Displays a list of the drawings in the specified file (without actually loading the file). You can see a list of drawings in files that have been read by clicking Right on Select Drawing in the command menu.
- Show Overview** Prompts for a record and displays its overview, showing how it fits into the overall document structure.
- Store into Register** Stores the selected entities into the chosen register. This command is normally accessed by clicking `c-m-Middle` on an entity, or Left on an empty register.
- The registers are named Register-1, Register-2, and Register-3, from left to right. You can create more registers than the three that are displayed by giving the name of a register from the keyboard to this command. A register name can be any token. Since these extra registers are not displayed, it helps if you give them symbolic names. For example, a register in which you are going to store a triangle and a square positioned like a house might be called House.
- Transform Entity** [Transform] Performs a general two-dimensional geometrical transformation on the selected entities. It pops up a menu of transformation specifications. The possible transformations are:



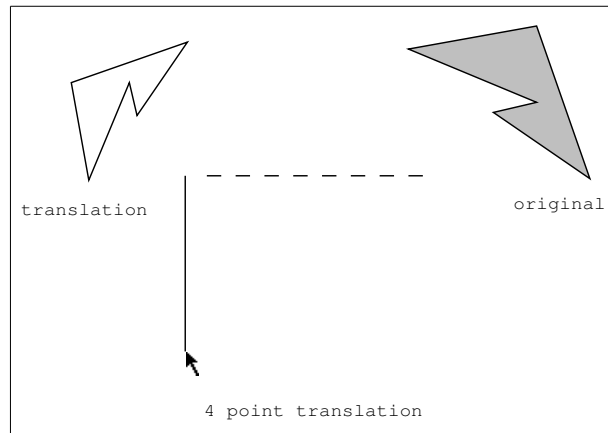
2 Point

Click Left on one point and release at another. The entity is translated by this vector.



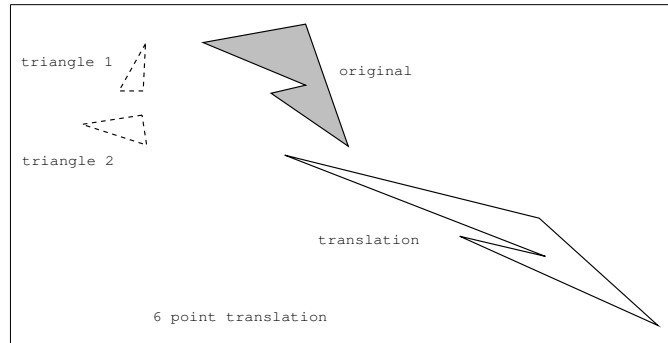
4 Point

Click Left on one point and release at another to give the starting vector. Repeat for the ending vector. The entity is Translated and rotated according to the difference.



6 Point

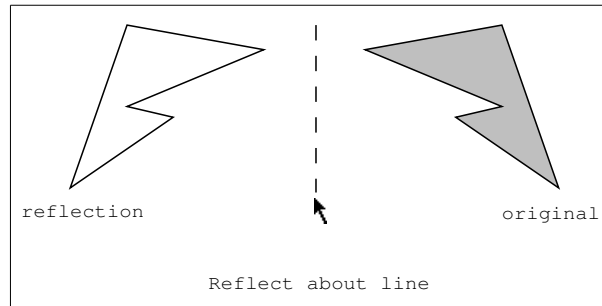
Input two triangles. The entity is transformed by the same combination of translation, rotation, reflection, and shearing as deforms the first triangle into the second.

**Move**

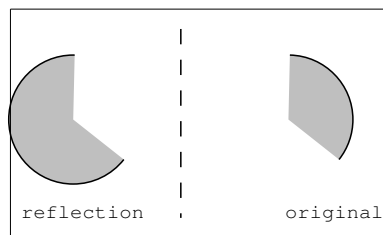
The mouse moves an image of the entities until you click someplace to place them down.

**Reflect About Line**

Click Left on one end of the line and release at the other. The entity is flipped to the other side of the line.



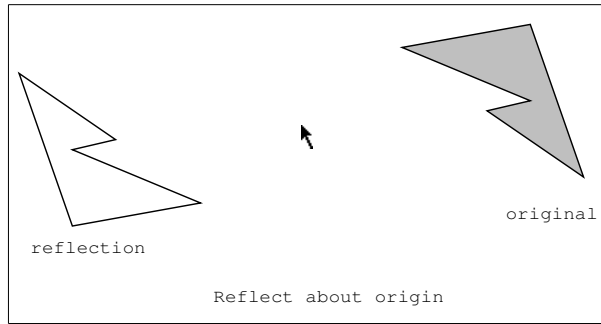
Note: if you reflect a circular arc, the arc is not flipped. The reflection is the complement of the arc (that is, the rest of the circle):



You can use Rotate to get the desired reflection.

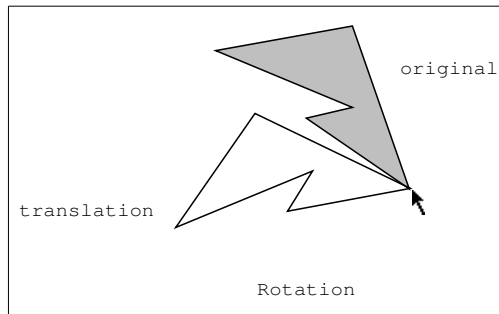
**Reflect About Origin**

Click Left on the desired center point.



Rotate

Click Left on a point to be the center of rotation and hold while moving a rotation radius to the desired angle.

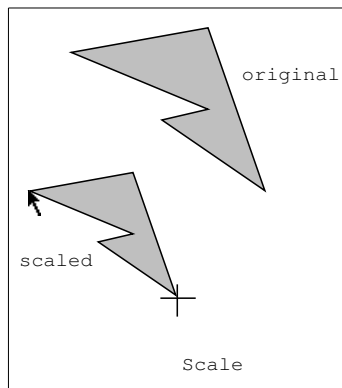


Rotate  $n$  degrees

Click Left on a point to be the center of rotation and enter a counterclockwise angle in degrees from the keyboard.

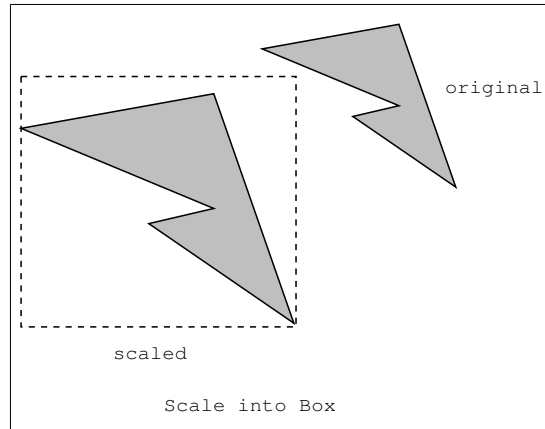
Scale

Click Left on the point to remain fixed and then click and hold another point to move. The entity is scaled evenly according to how much you move the second point.

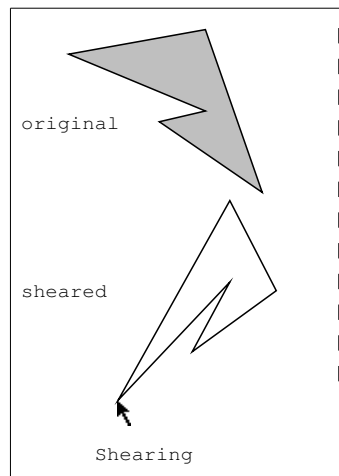


**Scale into Box**

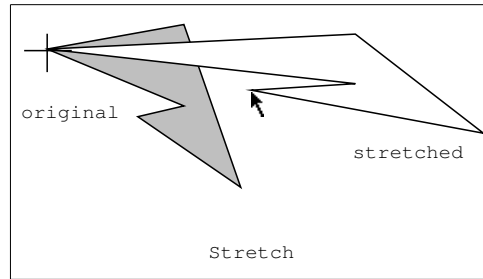
Click Left on the upper left and release at the lower right of a box. The entity is scaled evenly horizontally and vertically to fit into the box.

**Shear**

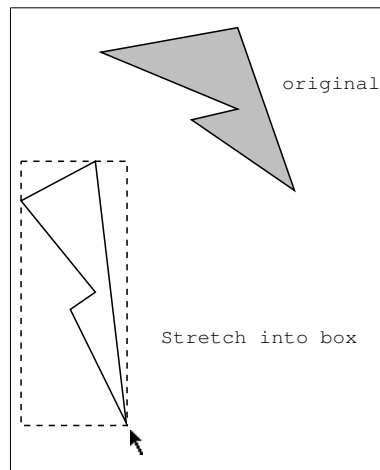
Click Left on the start of the line to remain fixed and release at other end. Then click and hold on a point to move. Release when the point is in the new position.

**Stretch**

Click Left on the point to remain fixed and then click and hold another point to move. The entity is deformed according to how much you move the second point.



**Stretch into Box** Click Left on the upper-left and release at the lower-right of a box. The entity is deformed horizontally and vertically to fit into box.



- Undo** [Undo] Undoes the last command that was saved in the undo history. The undo history is a tree. Branches are created when you go back and do something different.
- Undo Skip** [Skip] Moves forward in the undo history, like redo would, but does not actually do anything.
- Ungroup Entity** [Ungroup] Undoes the effect of a group command for the selected group. The grouped entities are split back into separate entities.
- Write File** [Write File] Writes the current drawing buffer out to a file. Actually, changes the file associated with the drawing to be the given file, and then writes the file to contain all the drawings associated with that file.

When you are working with multiple drawings in a file, use the "Set Drawing File Graphic Editor Command" to associate each drawing with the file, and then, when you are done edit-

ing all of the drawings for the file, use Write File. This prevents creating extra copies of the drawing file.

Note that the Graphic Editor drawing files are not displayed in the Concordia buffers list, so, when you logout or boot, you should remember to write the drawing file to save any unsaved drawings.

Also note that Symbolics Concordia automatically saves representations of drawings that are included in Concordia documents when you save the .sab file. So, unless you plan to edit these drawings later, there is no need to also save them in a drawing file.

**Zoom by Factor** [Zoom by Factor] Adjusts the scale of the drawing or image display by a specified factor. You are prompted for this factor, which can be any positive number. Numbers greater than 1 increase the scale, displaying more detail. Less than 1 does the inverse.

This adjustment is for display purposes only and does not affect the actual size of entities in the drawing or image.

**Zoom/Contract** [Zoom/Contract] Inputs a rectangle with the mouse and adjusts the scale and translation of the drawing display so that the whole previously displayed area scales down into the indicated rectangle.

This adjustment is for display purposes only and does not affect the actual size of entities in the drawing or image.

**Zoom/Expand** [Zoom/Expand] Adjusts the scale and translation of the drawing or image display so that that part of the drawing or image occupies the whole display area. Define the rectangular area to be zoomed by positioning the cursor at one corner and pressing and dragging the mouse to the opposite corner (release the mouse button to define the opposite corner).

This adjustment is for display purposes only and does not affect the actual size of entities in the drawing or image.

## 35. Using the Bitmap Editor

This section describes how to use the Bitmap Editor to edit bitmap images, such as

- Symbolics Concordia screen images
- Graphics imported as bitmap images into Symbolics Concordia from other systems
- Graphics scanned into Symbolics Concordia from printed documents

The most common use of the Bitmap Editor is to remove extra whitespace surrounding captured screen images ("Refit Bounding Box Bitmap Editor Command"). You can also use the Bitmap Editor to "clean up" scanned in pictures, or to add details (such as arrows or annotation) to captured screen images. For more information on how to edit a bitmap image, see the section "Editing Bitmap Images", page 326.

Usually you enter the Bitmap Editor with an existing bitmap image (such as the Named Image of a captured screen, or an image in an image file). Screen images captured during the current boot session are immediately accessible to the Bitmap Editor. To make images in an image file accessible to the Bitmap Editor use the "Read Image File Command".

### 35.1. Using the Bitmap Editor Window

The Bitmap Editor window looks like this:

Drawing Pane	The largest pane in the center is the drawing pane. It contains a grid of points forming an array of squares. The grid indicates the bit size. You can change the size of the grid using [Set Scale].
Preview Pane	The small pane in the upper left is shows the image that is in the drawing pane at its normal size (that is, unscaled).
Registers Pane	The boxes in the top right show the contents of the drawing registers. You can use these registers to temporarily store parts of the image for later use (for example, if you want to duplicate parts of your drawing).
Command Menu Pane	The menu is just below the Drawing Pane. It displays Bitmap Editor commands partitioned into functional groupings. You can click on these commands to initiate Bitmap Editor commands. (You can also enter Bitmap Editor commands by typing them directly in the Bitmap Editor Prompt Pane.) Note that

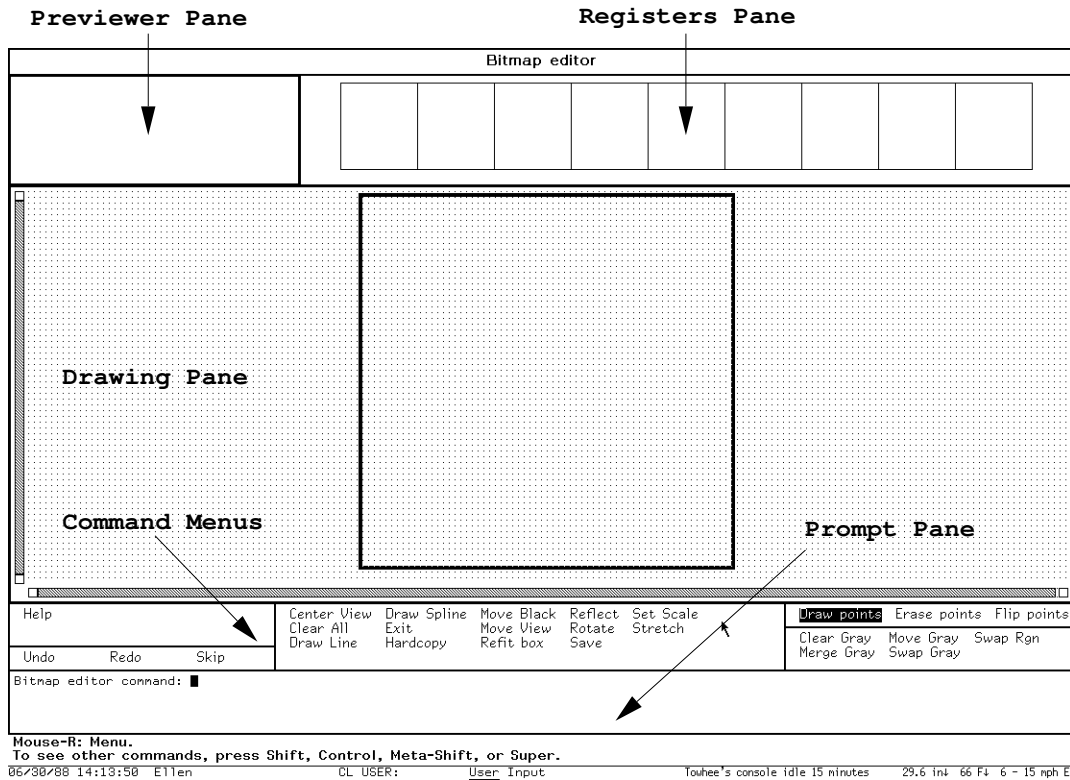


Figure 49. The Bitmap Editor Screen

only some of the Bitmap Editor commands can be selected via the menu. For a summary of all Bitmap Editor commands, see the section "Dictionary of Bitmap Editor Commands", page 327.

#### Prompt Pane

The small pane below the Command Menu Pane shows Bitmap Editor command prompts and responses. Enter Bitmap Editor commands at the following prompt:

Bitmap editor command:

Note that some commands may also require keyword arguments.

## 35.2. Entering Bitmap Editor Commands

You can enter Bitmap Editor commands by clicking Left on an option in a command menu or by entering the command directly using the keyboard. Note that all Bitmap Editor commands are not represented in Bitmap Editor command menus. Some commands must be entered via the keyboard. Note that some commands require additional keyword arguments.



The Prompt Pane at the bottom of the Bitmap Editor window shows Bitmap Editor command interaction. Bitmap Editor commands appear after the prompt

Bitmap editor command:

As with the Graphic Editor, you can click **Middle** on a command name in the help display or the menu and see documentation for the command.

You can click **Right** on a command name to see documentation on command arguments.

Pressing **HELP** shows possible command completions.

For a description of Bitmap Editor commands, see the section "Dictionary of Bitmap Editor Commands", page 327.

## 35.3. Bitmap Editor Basic Concepts

### 35.3.1. Drawing

Bitmap images are displayed and edited in the Bitmap Editor drawing pane. The most frequently used Bitmap Editor drawing commands appear in the command pane (just below the Bitmap Editor drawing pane). You can click on these commands or you can type commands directly using the keyboard. (For a complete list of Bitmap Editor commands, see the section "Dictionary of Bitmap Editor Commands", page 327.)

You can draw in one of three modes at any time, [Set Points], [Clear Points], or [Flip Points]. (The active draw mode is highlighted.)

When you click **Left** on a box in the drawing pane, that box is made black (set), or white (clear), or complemented (flip), according to the current draw mode. If you hold the left button down (that is, you do not release it after clicking **Left** on a box) and move it around, you set (that is, draw), clear, or complement all squares over which you pass. In this way, you can draw curves or pictures, fill in areas, or clear old mistakes.

Flip mode is particularly useful for final touch-ups, a click at a time, rather than drawing with the mouse button down. Since it changes any square you click on, it is most useful when you fix up single squares in the final stages of editing.

You can change the drawing mode either by selecting another draw mode by clicking on an item in the draw mode menu, or by clicking **Middle** on the drawing pane. Clicking **Middle** rotates through the possible draw modes. (You can also set the drawing mode by entering the Set Drawing Mode command at the Bitmap Editor Command prompt.)

When you draw with the mouse, the preview pane is not updated until you release the left button. (You might want to do this every now and then while drawing with the mouse, just to observe what you have in life-size, and then press the left button again, to continue drawing.)

You can "temporarily" change the draw mode while drawing by manipulating the `CONTROL` and `META` keys on the keyboard. If you hold down `CONTROL` alone while drawing, the temporary draw mode becomes [Clear Points] for as long as it is held down. Similarly, `META` alone sets up [Set Points] mode for as long as it is held down. `CONTROL` and `META` together temporarily put the mouse in *pass-over* mode, in which it makes no change to any squares it passes over. This technique is useful when you want to change the draw mode for just one or two squares.

### 35.3.2. Using the Gray and Black Planes to Edit Images

The image that you are editing in the Drawing Pane is in the *black* plane. The gray plane is a "shadow" behind the drawing pane that allows you to look at another image in addition to the one you are editing. You can exchange images between the black and gray planes using the "Swap Gray Bitmap Editor Command".

Frequently, the gray plane is used to serve as a guide for drawing the new image by holding an image that resembles (or has pieces of) the image being edited. At other times, the gray plane is used to hold a piece of an image, to be merged later into the black plane.

The most common way of putting drawings into the gray plane to move the black plane into it using [Swap Gray]. The image (or piece of an image) in the gray plane shows up in light gray in the drawing pane. Bits that are on in both the gray plane and the black plane, appear as dark gray squares.

The following Bitmap Editor commands allow you to manipulate gray and black plane images:

Clear All Bitmap Editor Command

Erases both the black and gray planes.

Clear Gray Bitmap Editor Command

Erases the gray plane.

Merge Gray Bitmap Editor Command

Copies the gray plane into the black plane without erasing either.

Move Black Bitmap Editor Command

Adjusts the position of drawn points relative to the bounding box.

Move Gray Bitmap Editor Command

Adjusts the position of the gray points relative to the bounding box.

Retrieve Register Into Black Bitmap Editor Command

Copies the contents of the given register into the black plane.

Retrieve Register Into Gray Bitmap Editor Command

Copies the contents of the given register into the gray plane.

**Store Black Into Register Bitmap Editor Command**

Copies the black plane into the given register, erasing what was previously there.

**Store Gray Into Register Bitmap Editor Command**

Copies the gray plane into the given register, erasing what was previously there.

**Swap Gray Bitmap Editor Command**

Exchanges the black and gray planes.

**Swap Region Bitmap Editor Command**

Exchanges a subset of the blank and gray planes contains in a rectangle specified with the mouse.

**35.3.3. Using Registers to Copy Images**

You can use Bitmap Editor registers to save portions of bitmap images for later use. Registers are particularly useful when you want to make duplicate (or slightly modified) copies of portions of the drawing.

Bitmap Editor registers appear in the upper-right portion of the screen. Registers are mouse-sensitive, and grow a thick border when you move the mouse over them. Clicking Left on an empty register (one that looks blank) saves the black plane of the drawing in that register. Clicking Left on a register that is not empty, transfers the contents of the register to the black plane of the drawing.

Use the following commands to manipulate Bitmap Editor drawing registers:

**Clear Register Bitmap Editor Command**

Erases the given register, making it so that clicking Left on it will store into it rather than retrieving.

**Retrieve Register Into Black Bitmap Editor Command**

Copies the contents of the given register into the black plane.

**Retrieve Register Into Gray Bitmap Editor Command**

Copies the contents of the given register into the gray plane.

**Store Black Into Register Bitmap Editor Command**

Copies the black plane into the given register, erasing what was previously there.

**Store Gray Into Register Bitmap Editor Command**

Copies the gray plane into the given register, erasing what was previously there.

**35.3.4. Viewing Bitmap Images**

Use the following commands to change the view (display) of Bitmap Editor images:

**Center View Bitmap Editor Command**

Adjusts the view so that the center is in the center of the window.

**Center View Bitmap Editor Command**

Adjusts the view so that the center is in the center of the window.

**Zoom by Factor Bitmap Editor Command**

Adjusts the scale of the drawing or image display by a specified factor.

**Zoom/Expand Bitmap Editor Command**

Adjusts the scale and translation of the drawing or image display so that that part of the drawing or image occupies the whole display area.

## 35.4. Using Screen Images as Illustrations

To use a screen image as an illustration in a Symbolics Concordia document

1. Capture the image using `FUNCTION Ø Ø` (and specify the name of the new image). For more information, see the section "Capturing Screen Images", page 322.
2. Edit the bitmap image using the Bitmap Editor (if necessary).
3. Insert the captured image in a Graphic Editor drawing using the "Add Image Graphic Editor Command". You can also use the Graphic Editor to add other drawing entities (such as arrows or text).
4. Include the drawing in a Symbolics Concordia document using `m-x` "Create Picture".

### 35.4.1. Capturing Screen Images

`FUNCTION Ø Ø` sends a full screen image to a printer.

Use `FUNCTION Ø Ø` (and specify a Named Image) to capture a screen image for inclusion in a document.

If you need to edit an image, use the Bitmap Editor ("Edit Image Bitmap Editor Command"). If you want to include the image in a Symbolics Concordia document, use the Graphic Editor ("Add Image Graphic Editor Command").

Note that, although you cannot edit a bitmap image in the Graphic Editor, you can combine it with other graphic elements (such as arrows and labels).

If you need to modify bitmap image that is in a Graphic Editor drawing, use the "Copy Drawing to Image Graphic Editor Command" to make a bitmap image of the drawing, and then use the "Edit Image Bitmap Editor Command" to make the image accessible for editing.

### Overview of the Screen Hardcopy Menu

When you enter FUNCTION 0 0, the Screen Hardcopy menu appears. You can click on menu options to define the characteristics of the resulting bitmap image (such as the region of the screen to capture, or the image destination). Note that some menu options prompt for additional information (for example, you can supply the image name for a bitmap to be saved in a [Named Image]).

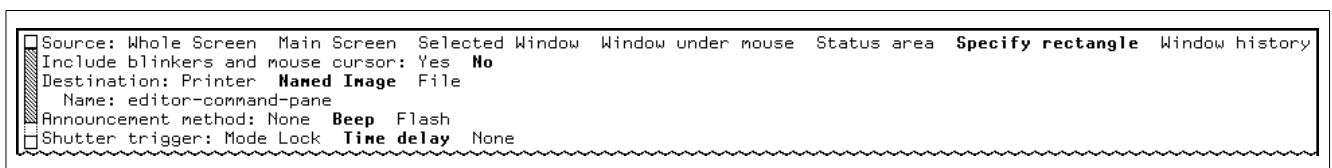


Figure 50. FUNCTION 0 0

Destination: options define where to put the captured bitmap image. Selecting [Named Image] makes a bitmap image accessible to the Bitmap Editor for editing ("Edit Image Bitmap Editor Command", or to the Graphic Editor for including in a Symbolics Concordia document ("Add Image Graphic Editor Command").

When you choose the [Named Image] destination, you can override the default image name by positioning the mouse cursor to highlight the default name (after the "Name:" field), click on the name, and type the new name. Note that image names must not contain blanks. (Remember to press RETURN when you are done.)

Source: options identify the area of the screen to capture:

- |                        |  |
|------------------------|--|
| Whole Screen           | A bitmap image of the entire screen  |
|                        | Note that a full-screen bitmap image includes a border around the actual screen image. If you do not want this extra white-space around the image, select the Named Image option and edit the bitmap image using the "Refit Bounding Box Bitmap Editor Command". |
| Main Screen            | All but the status area  |
| Selected Window        | The window of the selected activity (for example, Symbolics Concordia)   |
| Window Under the Mouse | The window under the mouse cursor  |

	Use this option when you cannot use the mouse to specify a portion of the screen (for instance, to take a snapshot of a pop-up menu).
Status Area	The window at the bottom of the screen (including the mouse information line(s) )
Specify Rectangle	Specify the area of the screen to be captured using the mouse
Window History	The history of the specified window
Shutter Trigger:	
Mode Lock	Captures the image when the MODE LOCK key is pressed. This allows you to <i>set up</i> the screen a certain way, perhaps with a menu showing, or the mouse cursor in a certain place.
Time Delay	Allows you to specify a time delay before capturing the image. The default is 5 seconds.
None	Captures the image as soon as you click on [Done]. This is the default.

### Example of Capturing a Specified Rectangle of a Screen

This example shows how to use [Specify Rectangle] to capture the display of a font after a Show Font command.

To capture the font display area of the screen:

1. In a Lisp Listener, type Show Font Tiny.
2. Position your mouse cursor just above and to the left of the beginning of the display of the font.
3. Press FUNCTION  $\square$   $\square$  and click on [Specify Rectangle] and [Named Image]. See Figure 50.
4. To specify a new image name, position the mouse over the image name (following "Name:"). The default image name is highlighted by a rectangle. Click on the highlighted rectangle and type the new name to override the default. The selected option will be **bolded**.
5. Click on [Done] to capture the image and to exit from the Screen Hardcopy menu. A rectangle appears with its upper-left corner at the current mouse cursor location. Move the mouse cursor to define the lower-right corner of the rectangle (making sure that the lower-right corner is to the right of the upper-left corner).

You can relocate the upper-left corner of the rectangle by holding down the SHIFT key and moving the mouse. When the rectangle is positioned where you want it, release the SHIFT key.



3. Use the Add Image command (with the named image and a scale factor) to include the bitmap image in the current Graphic Editor drawing. Move the mouse cursor and click Left to position the bitmap image in the drawing pane.

Note that in general it is better to accept the default scale factor (1 for full size). You can use the `PictureScale` environment attribute to resize your drawing when you insert the final picture in your document. (See the section "Specifying the Size of Pictures", page 180.)

Graphic Editor command: `Add Image show-font-display`

The image "show-font-display" is then inserted into the current drawing at the mouse cursor location.

4. Use Graphic Editor commands to add drawing entities (such as arrows and comments) as needed.
5. Click on [Done] to exit the Graphic Editor activity.

### 35.4.3. Editing Bitmap Images

To edit a bitmap image:

1. Press `SELECT }` (or type `Select Activity Bitmap Editor` in a Lisp Listener) to initiate the Bitmap Editor activity (the current image is the default).
2. Select the appropriate bitmap image by entering the "Edit Image Bitmap Editor Command" and specifying the image name.

There are several ways to make a bitmap image accessible to the Bitmap Editor.

- You can use the "Read Image File Command" if the bitmap image is in an image file.
  - You can capture and name a screen image (using `FUNCTION Ø Ø`, and specifying a destination of `Named Image` with an *image name*).
  - You can use "Copy Drawing to Image Graphic Editor Command" for a Graphic Editor drawing.
3. Use Bitmap Editor commands to modify the image ( see the section "Dictionary of Bitmap Editor Commands", page 327).
  4. To create a named image that is accessible to the "Add Image Graphic Editor Command" click Left on [Save] in the menu. (Note that this command does *not* save the image.)



5. If you want to save the edited image use the "Write Image File Command" .

Note that bitmap images take a lot of space, and bitmap images in Symbolics Concordia illustrations are automatically saved with the .sab file. Save bitmap images only if they can't be easily reproduced, or if you think you may need to edit them again in the future.

6. Click on [Done] in the menu. This allows reuse of the Bitmap Editor activity for your next image. If you have not finished editing the image, exit by selecting a new activity.

For information on how to insert a drawing into a document, see the section "Including a Graphic Editor Drawing in a Document", page 270.

Note that when you save a Symbolics Concordia document with a picture in it, a bitmap image of the picture is also saved. So, if you need to conserve space (or if you don't plan to edit the drawing in the future), you might not want to save the image in a separate image file.

### 35.5. Dictionary of Bitmap Editor Commands

Center View	Adjusts the view so that the center is in the center of the window.
Clear All	[Clear All] Erases both the black and gray planes.
Clear Gray	Erases the gray plane.
Clear Register	Erases the given register, making it so that clicking Left on it will store into it rather than retrieving.
Clear Undo History	Clears everything out of the undo history. The operations in the undo history then become accessible to the garbage collector.
Command Help	See "Help Bitmap Editor Command"
Create Image	Creates an image.
Done	[Done] Exits the program, returning you to your previous context.
Draw Line	[Draw Line] Draws a line between two given points.
Draw Point	Draws a point in the current drawing mode when you click with the mouse.  Holding down the mouse button after clicking draws successive points as the mouse moves. Pressing the CONTROL key while drawing inverts the drawing mode.

You can "temporarily" change the draw mode while drawing by manipulating the `CONTROL` and `META` keys on the keyboard. If you hold down `CONTROL` alone while drawing, the temporary draw mode becomes [Clear Points] for as long as it is held down. Similarly, `META` alone sets up [Set Points] mode for as long as it is held down. `CONTROL` and `META` together temporarily put the mouse in *pass-over* mode, in which it makes no change to any squares it passes over. This technique is useful when you want to change the draw mode for just one or two squares.

Draw Polygon	Draws a filled polygon with corners at the given points.
Draw Rectangle	[Draw Rect] Draws a filled rectangle with corners at two given points.
Draw Spline	[Draw Spline] Draws a cubic spline with the given control points.
Edit Image	[Edit] Places a bitmap image into the Bitmap Editor for editing. It prompts for a named image, with the default being the current image.
Flood Region	[Flood] Complements a contiguous area of pixels of the same color given a seed point.
Hardcopy Image	Hardcopies the current image.
Help	The Help command prints out help on other commands, using the standard documentation database and formatter.  After clicking on the Help command, you can enter a topic, such as a command name, from the keyboard, or select a menu item from the editor's command menus.
Merge Gray	Copies the gray plane into the black plane without erasing either.
Move Black	[Move Black] Adjusts the position of drawn points relative to the bounding box.
Move Bounding Box	Moves the bounding box. Normally accessed by clicking on the bounding box or one of its edges.
Move Gray	Adjusts the position of the gray points relative to the bounding box.
Move View	[Move View] Adjusts the position of the view window.
Read Image File	Read Image File <i>pathname format keywords</i>

Reads a picture or bitmap image from a file.

<i>pathname</i>	The pathname of the file from which to read an image.
<i>format</i>	The format of the data in the file. These are the possible image file formats:
BIN	Symbolics binary file (data only, compatible with all architectures)
Compact Bitmap	X windows "compact" bitmap
EPS/Macintosh	Postscript in the data fork, PICT in the resource fork
Gem IMG	Used by Ventura Publisher
GIF	CompuServe Graphics Interchange Format
Macintosh	Macintosh based on type
MacPaint	MacPaint canvas (fixed size)
PC Paintbrush	PC Paintbrush format
PICT	Macintosh picture record
Portable Bitmap	X windows "portable" bitmap (1's and 0's)
PostScript	Reads any PostScript images, writes EPSF compatible
TIFF	Tagged Image File Format
Truevision	Truevision product family format
Utah RLE	University of Utah format Run-length-encoding
X Bitmap	X windows bitmap (C code)
X Window Dump	X v.11 window dump
Xim	X Image

	<i>keywords</i>	:More Processing, :Output Destination, :Rotate, :Trim
	:More Processing	{Default, Yes, No} Controls whether <b>More</b> processing at end of page is enabled during output to interactive streams. The default is Default. If No, output from this command is not subject to <b>More</b> processing. If Default, output from this command is subject to the prevailing setting of <b>More</b> processing for the window. If Yes, output from this command is subject to <b>More</b> processing unless it was disabled globally (see the section "FUNCTION M" in <i>General Handbook</i> ).
	:Output Destination	{Buffer, File, Kill Ring, None, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream <b>standard-output</b> .
	:Rotate	{Yes, No} Rotate the image 90 degrees (useful for MacPaint screen images).
	:Trim	{Yes, No} Return the smallest image necessary to store just the ink.
Redo		Redoes the last command that was undone via the Undo command.  It is possible to have more than one command that got done after the command that you have undone back to. In this case, a menu is popped up. Newer commands are at the bottom.
Refit Bounding Box		Adjusts the edges of the bounding box to exactly fit the drawn area. This is useful for removing excess whitespace around an image, for example, the border on a full screen image.  Note that this command only removes white borders. If you want to use Refit Boundin Box on an image with a background that is not white (the default), you must first reset the background to white and remake the image.
Reflect Diagonally		Reflects the black plane about a diagonal line passing through a given point.
Reflect Horizontally	[Reflect]	Reflects the black plane about a given horizontal line.
Reflect Vertically		Reflects the black plane about a given vertical line.
Refresh	[Refresh]	Redraws the drawing or image display.

Retrieve Register Into Black	Copies the contents of the given register into the black plane.						
Retrieve Register Into Gray	Copies the contents of the given register into the gray plane.						
Rotate	[Rotate] Rotates the black plane about a given point.						
Save Image	[Save] Replaces the raster being edited with the edited image.						
Set Drawing Mode	[Draw] [Erase] [Flip] Changes the current drawing mode. Usually accessed by clicking on one of the drawing modes in the drawing modes menu. The modes are: <table> <tr> <td>Draw Points</td> <td>Click to insert a black bit.</td> </tr> <tr> <td>Erase Points</td> <td>Click to remove a black bit.</td> </tr> <tr> <td>Flip Points</td> <td>Click to change a black bit to white and vice versa.</td> </tr> </table>	Draw Points	Click to insert a black bit.	Erase Points	Click to remove a black bit.	Flip Points	Click to change a black bit to white and vice versa.
Draw Points	Click to insert a black bit.						
Erase Points	Click to remove a black bit.						
Flip Points	Click to change a black bit to white and vice versa.						
Set Scale	[Set Scale] Changes the displayed scale in pixels.						
Snap Bounding Box [⇧-Middle]	Redefines the bounding box. You use ⇧-Middle with the cursor near the bounding box to drag the box edge to a new position. You can also move the whole box. Look at the Mouse Documentation line to see how the gesture will affect the box.						
Store Black Into Register	Copies the black plane into the given register, erasing what was previously there.						
Store Gray Into Register	Copies the gray plane into the given register, erasing what was previously there.						
Stretch Column	[Stretch] Makes copies of a given column to fill up a given space.						
Stretch Row	Makes copies of a given row to fill up a given space.						
Swap Gray	Exchanges the black and gray planes.						
Swap Region	Exchanges a subset of the black and gray planes contained in a rectangle specified with the mouse.						
Zoom by Factor	[Zoom by Factor] Adjusts the scale of the drawing or image display by a specified factor. You are prompted for this factor,						

which can be any positive number. Numbers greater than 1 increase the scale, displaying more detail. Less than 1 does the inverse.

**Zoom/Expand** [Zoom/Expand] Adjusts the scale and translation of the drawing or image display so that that part of the drawing or image occupies the whole display area. Define the rectangular area to be zoomed by positioning the cursor at one corner and pressing and dragging the mouse to the opposite corner (release the mouse button to define the opposite corner).

**Undo** Undoes the last command that was saved in the undo history. The undo history is a tree. Branches are created when you go back and do something different.

**Undo Skip** Moves forward in the undo history, like redo would, but does not actually do anything.

## Write Image File

Write Image File *image pathname format keywords*

Saves an image in a file.

<i>image</i>	The image to be written to a file.
<i>pathname</i>	The file in which to save the image.
<i>format</i>	The format of the data in the file. The possible image file formats are:
BIN	Symbolics binary file (data only, compatible with all architectures)
Compact Bitmap	X windows "compact" bitmap
EPS/Macintosh	Postscript in the data fork, PICT in the resource fork
MacPaint	MacPaint canvas (fixed size)
PICT	Macintosh picture record
Portable Bitmap	X windows "portable" bitmap (1's and 0's)
PostScript	Reads any PostScript images, writes EPSF compatible

TIFF	Tagged Image File Format
X Bitmap	X windows bitmap (C code)

*keywords*

:More Processing, :Output Destination

:More Processing {Default, Yes, No} Controls whether **\*\*More\*\*** processing at end of page is enabled during output to interactive streams. The default is Default. If No, output from this command is not subject to **\*\*More\*\*** processing. If Default, output from this command is subject to the prevailing setting of **\*\*More\*\*** processing for the window. If Yes, output from this command is subject to **\*\*More\*\*** processing unless it was disabled globally (see the section "FUNCTION M" in *Genera Handbook*).

:Output Destination {Buffer, File, Kill Ring, None, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream **\*standard-output\***.

Zoom by Factor [Zoom by Factor] Adjusts the scale of the drawing or image display by a specified factor. You are prompted for this factor, which can be any positive number. Numbers greater than 1 increase the scale, displaying more detail. Less than 1 does the inverse.

Zoom/Expand [Zoom/Expand] Adjusts the scale and translation of the drawing or image display so that that part of the drawing or image occupies the whole display area. Define the rectangular area to be zoomed by positioning the cursor at one corner and pressing and dragging the mouse to the opposite corner (release the mouse button to define the opposite corner).





## 36. Using the Stipple Editor

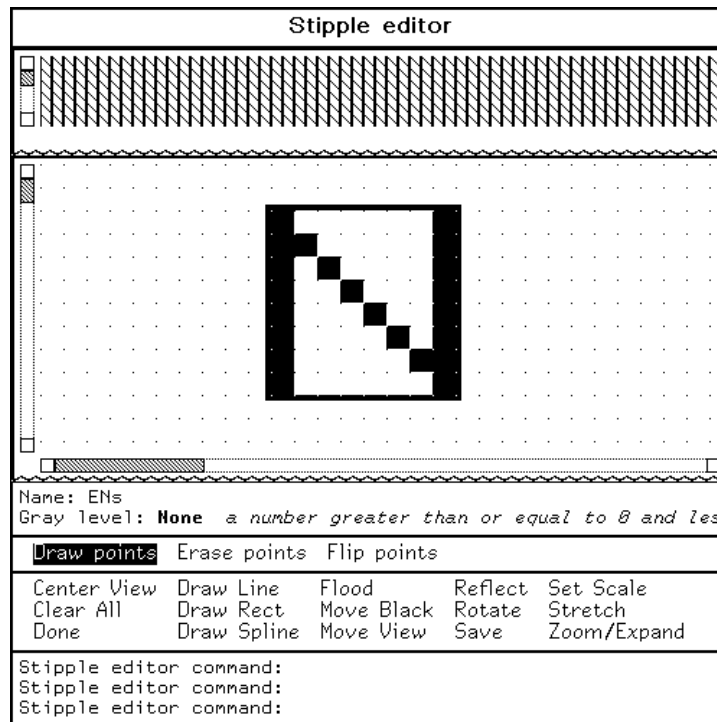
The Graphic Editor provides a default set of stipple patterns that you can use to fill Graphic Editor shapes ( see the section "Filling a Shape Entity with a Pattern", page 285).

You can use the Stipple Editor to define new patterns for filling Graphic Editor shape entities.

Use the "Define New Pattern Graphic Editor Command" to create a new stipple pattern. This command selects the Stipple Editor activity (you can also press **SE-LECT** |). After you have created the new stipple pattern, you can use it to fill a selected shape entity by clicking on the sample pattern in the Graphic Editor Change entity options menu ( see the section "Filling a Shape Entity with a Pattern", page 285). Note that new stipple patterns exist only in the Graphic Editor drawings where they are used. You cannot save stipple patterns.

### 36.1. Creating a New Stipple Pattern

Use the Stipple Editor window to define new stipple patterns. This window shows a new stipple pattern called "ENs".



Note that the top pane shows how your stipple looks when you use it to fill a shape. This display changes as you edit the stipple pattern.

If you [Save] the pattern, you can use it to fill Graphic Editor shapes by selecting the sample pattern from the [Change] menu.

For a detailed description of Stipple Editor Commands, see the section "Dictionary of Stipple Editor Commands", page 336. You can also see the section "Using the Bitmap Editor", page 317 for more information on basic bitmap editing concepts.

## 36.2. Techniques for Using the Stipple Editor

- You can extend the stipple Bounding Box by highlighting a border and dragging it to the new position. This enlarges the area you can use to define the new stipple pattern.
- Note that you cannot save stipple patterns across boot sessions. The new pattern exists only in the drawings that use it. [Save] allows you to select the new stipple pattern using the [Change] menu.

## 36.3. Dictionary of Stipple Editor Commands

Center View	Adjusts the view so that the center is in the center of the window.
Clear All	[Clear All] Erases both the black and gray planes.
Clear Gray	Erases the gray plane.
Clear Undo History	Clears everything out of the undo history. The operations in the undo history then become accessible to the garbage collector.
Command Help	[Help] See "Help Stipple Editor Command"
Draw Line	[Draw Line] Draws a line between two given points.
Draw Point	<p>Draws a point in the current drawing mode when you click with the mouse.</p> <p>Holding down the mouse button after clicking draws successive points as the mouse moves. Pressing the CONTROL key while drawing inverts the drawing mode.</p> <p>You can "temporarily" change the draw mode while drawing by manipulating the CONTROL and META keys on the keyboard. If you hold down CONTROL alone while drawing, the temporary draw mode becomes [Clear Points] for as long as it is held down. Similarly, META alone sets up [Set Points] mode for as long as it is held down. CONTROL and META together temporarily</p>

	put the mouse in <i>pass-over</i> mode, in which it makes no change to any squares it passes over. This technique is useful when you want to change the draw mode for just one or two squares.
Draw Polygon	Draws a filled polygon with corners at the given points.
Draw Rectangle	[Draw Rect] Draws a filled rectangle with corners at two given points.
Draw Spline	[Draw Spline] Draws a cubic spline with the given control points.
Exit Stipple Editor	[Done] Exits the program, returning you to your previous context.
Flood Region	[Flood] Complements a contiguous area of pixels of the same color given a seed point.
Help	The Help command prints out help on other commands, using the standard documentation database and formatter.  After clicking on the Help command, you can enter a topic, such as a command name, from the keyboard, or select a menu item from the editor's command menus.
Merge Gray	Copies the gray plane into the black plane without erasing either.
Move Bounding Box	Moves the bounding box. Normally accessed by clicking on the bounding box or one of its edges.
Move Gray	Adjusts the position of the gray points relative to the bounding box.
Move View	[Move View] Adjusts the position of the view window.
Redo	Redoes the last command that was undone via the Undo command.  It is possible to have more than one command that got done after the command that you have undone back to. In this case, a menu is popped up. Newer commands are at the bottom.
Reflect Diagonally	Reflects the black plane about a diagonal line passing through a given point.
Reflect Horizontally	[Reflect] Reflects the black plane about a given horizontal line.
Reflect Vertically	Reflects the black plane about a given vertical line.
Refresh	[Refresh] Redraws the drawing or image display.

Rotate	[Rotate] Rotates the black plane about a given point.						
Save Stipple Editor	[Save] Saves the new stipple pattern. The pattern name and sample patch will appear in the sample pattern menu of the Define New Pattern Graphic Editor command menu (or when you click on Patterned in the Change Entity menu).  Note that when you reboot your system, the new stipple pattern is retained in the Graphic Editor drawing, but not retained in the sample pattern menu.						
Set Drawing Mode	[Draw] [Erase] [Flip] Changes the current drawing mode. Usually accessed by clicking on one of the drawing modes in the drawing modes menu.  The modes are: <table> <tr> <td>Draw Points</td> <td>Click to insert a black bit.</td> </tr> <tr> <td>Erase Points</td> <td>Click to remove a black bit.</td> </tr> <tr> <td>Flip Points</td> <td>Click to change a black bit to white and vice versa.</td> </tr> </table>	Draw Points	Click to insert a black bit.	Erase Points	Click to remove a black bit.	Flip Points	Click to change a black bit to white and vice versa.
Draw Points	Click to insert a black bit.						
Erase Points	Click to remove a black bit.						
Flip Points	Click to change a black bit to white and vice versa.						
Set Pixel	Sets a specified pixel to a value (1 for On, or 0 for Off). You can specify Foreground or Background and Draw, Erase, or Flip.						
Set Scale	[Set Scale] Changes the displayed scale in pixels.						
Snap Bounding Box	[ $\text{⇧}$ -Middle] Redefines the bounding box.  You use $\text{⇧}$ -Middle with the cursor near the bounding box to drag the box edge to a new position. You can also move the whole box. Look at the Mouse Documentation line to see how the gesture will affect the box.						
Stretch Column	[Stretch] Makes copies of a given column to fill up a given space.						
Stretch Row	Makes copies of a given row to fill up a given space.						
Swap Gray	Exchanges the black and gray planes.						
Swap Region	Exchanges a subset of the blank and gray planes contained in a rectangle specified with the mouse.						
Undo	Undoes the last command that was saved in the undo history. The undo history is a tree. Branches are created when you go back and do something different.						
Undo Skip	Moves forward in the undo history, like redo would, but does not actually do anything.						

- Zoom by Factor** [Zoom by Factor] Adjusts the scale of the drawing or image display by a specified factor. You are prompted for this factor, which can be any positive number. Numbers greater than 1 increase the scale, displaying more detail. Less than 1 does the inverse.
- Zoom/Expand** [Zoom/Expand] Adjusts the scale and translation of the drawing or image display so that that part of the drawing or image occupies the whole display area. Define the rectangular area to be zoomed by positioning the cursor at one corner and pressing and dragging the mouse to the opposite corner (release the mouse button to define the opposite corner).



## **PART VI.**

### **CONVERTING EXISTING DOCUMENTATION TO CONCORDIA**





## 37. Introduction

What kinds of documents should be converted to Symbolics Concordia? Ideal candidates are large documents that require frequent updating and whose individual sections can be reused in other documents. Any document that you want to view online in Document Examiner must be converted to Symbolics Concordia.

Symbolics Concordia provides a set of tools to help with importing and converting documents written in other formats:

<b>Format</b>	<b>Concordia Command</b>
Scribe	m-x "Parse and Replace Region"
Text	m-x "Convert Flat Text to Record Command"
	m-x "Create Record"
	m-x "Create Link and Record"
	m-x "Create Link"
	m-x "Set Record Name Prefix Command"
	m-x "Set Record Name Suffix Command"
	m-x "Clear Record Name Prefix Command"
	m-x "Clear Record Name Suffix Command"
Graphics	m-x "Read Image File Command"
	m-x "Write Image File Command"



## 38. Converting Scribe Documents to Symbolics Concordia

To convert Scribe files (with an .mss file type), to Symbolics Concordia files (with a .sab file type), proceed as follows:

1. Create a Symbolics Concordia file with `c-X c-F`, for example:

```
c-X c-F my-file.sab RETURN
```

Create the initial top-level record for the new file with `Create Record`. The name of the top-level record should be the name of your document.

2. Yank the contents of the Scribe file into this new record.
3. Make the text suitable for online lookup:
  - Use `Create Record` to modularize the text. Place the first chapter in its own record and insert an include link in the top-level record. Break the record into smaller units; for example, put each section in its own record. Continue this process until the text is sufficiently modular. The acid test: Since each record is retrievable via `Document Examiner`, each record should make sense on its own, without reference to sections that precede or follow it in the printed document.
  - Add keywords (similar to index entries) to the `Keywords` field. `Document Examiner` scans all keywords and topic names in the database to process `Show Candidates` queries. So take the reader's point of view: choose keywords that you think he or she might use to look up information.
  - Edit record names: names should be unique, fairly specific, and informative. "Introduction to Foo" is preferable to "Introduction". "Helpful Hints" is useless, since readers are not likely to search for information on such a vague topic. "Hints for Getting Around Foo" is better.

Mark the file as a region and use `m-X "Parse and Replace Region"` to turn all the `@` formatting markup into Symbolics Concordia markup.

4. If there are specially defined macros used in the file, you have to deal with those individually (that is, look at the formatting intent of the macro and choose the appropriate Symbolics Concordia format environment).
5. Format the top-level record online (`s-P`) to look for formatting problems: bad line breaks, too little white space around formatted text, and the like. Add more white space (a few carriage returns) around markup diagrams. This usually fixes things.



## 39. Converting Text Files to Symbolics Concordia

1. Create a Symbolics Concordia file with `c-X c-F`, for example:

```
c-X c-F my-file.sab RETURN
```

Make the initial top-level record for the new file with Create Record. The name of the top-level record should be the name of your document.

2. Yank the contents of the text file containing your document into this new record.
3. To modularize the document and to make it suitable for online lookup, you can use `m-X "Create Record"`, `m-X "Create Link"`, and/or `m-X "Create Link and Record"`.
4. If you are creating many new records from a text file, you can also use the `m-X "Convert Flat Text to Record Command"`, which makes new Symbolics Concordia records from marked sections in flat text (the title of the new record is the first line of the marked region). To assure that the new topic names are unique you can also use the following commands to define record name prefixes or suffixes for the new records:

### Clear Record Name Prefix Command

Clears the record name prefix prepended to newly created records.

### Clear Record Name Suffix Command

Clears the record name suffix appended to newly created records.

### Set Record Name Prefix Command

Sets the record name prefix prepended to newly created records.

### Set Record Name Suffix Command

Sets the record name suffix appended to newly created records.

5. Use Include links to create the document topic hierarchy.
  - Place the first chapter in its own record and insert an Include link in the top-level record. Break the record into smaller units; for example, put each section in its own record. Continue this process until the text is sufficiently modular. The acid test: Since each record is retrievable via Document Examiner, each record should make sense on its own, without reference to sections that precede or follow it in the printed document.

- Add keywords (similar to index entries) to the Keywords field. Document Examiner scans all keywords and topic names in the database to process Show Candidates queries. So take the reader's point of view: choose keywords that you think he or she might use to look up information.
  - Edit record names: names should be unique, fairly specific, and informative. "Introduction to Foo" is preferable to "Introduction". "Helpful Hints" is useless, since readers are not likely to search for information on such a vague topic. "Hints for Getting Around Foo" is better.
6. If there is @ markup in the file, you can mark the file as a region and use `m-x "Parse and Replace Region"` to convert the @ formatting markup into Symbolics Concordia markup.
  7. You can use the Symbolics Concordia string search and replace functions to replace or remove source document text formatting macros or markup commands. Examine the intent of the command, and create the appropriate Symbolics Concordia environment or command.
  8. Format the top-level record online (`s-P`) to look for formatting problems: bad line breaks, too little white space around formatted text, and the like. Add more white space (a few carriage returns) around markup diagrams. This usually fixes things.

## 40. Converting Graphics to Symbolics Concordia

You can use the "Read Image File Command" to make existing images accessible to the Graphic Editor ( "Add Image Graphic Editor Command"). You can use the "Write Image File Command" to write an image file.

For more information on how to use the imported images in Symbolics Concordia documents, see the section "Using Screen Images as Illustrations", page 322.

### Read Image File Command

Read Image File *pathname format keywords*

Reads a picture or bitmap image from a file.

<i>pathname</i>	The pathname of the file from which to read an image.
<i>format</i>	The format of the data in the file. These are the possible image file formats:
BIN	Symbolics binary file (data only, compatible with all architectures)
Compact Bitmap	X windows "compact" bitmap
EPS/Macintosh	Postscript in the data fork, PICT in the resource fork
Gem IMG	Used by Ventura Publisher
GIF	CompuServe Graphics Interchange Format
Macintosh	Macintosh based on type
MacPaint	MacPaint canvas (fixed size)
PC Paintbrush	PC Paintbrush format
PICT	Macintosh picture record
Portable Bitmap	X windows "portable" bitmap (1's and 0's)
PostScript	Reads any PostScript images, writes EPSF compatible
TIFF	Tagged Image File Format
Truevision	Truevision product family format

## Utah RLE

University of Utah format Run-length-encoding

X Bitmap X windows bitmap (C code)

X Window Dump

X v.11 window dump

Xim X Image

*keywords*

:More Processing, :Output Destination, :Rotate, :Trim

- :More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during output to interactive streams. The default is Default. If No, output from this command is not subject to **More** processing. If Default, output from this command is subject to the prevailing setting of **More** processing for the window. If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M" in *Genera Handbook*).
- :Output Destination {Buffer, File, Kill Ring, None, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream **standard-output**.
- :Rotate {Yes, No} Rotate the image 90 degrees (useful for MacPaint screen images).
- :Trim {Yes, No} Return the smallest image necessary to store just the ink.



## 41. Quick Conversion for Producing Printed Books

If you have an existing document that you want to produce as a book using the Symbolics Concordia formatter, you can do a *quick and dirty* job of it, especially if the document is written in Scribe or in Zmacs format file commands.

1. Create a top level record with the same name as the document.
2. Insert the entire text of the document in this record.
3. If the document contains Scribe or Zmacs format file commands, mark the entire record and use `m-x` "Parse and Replace Region" to change the markup to Symbolics Concordia markup.

Each `@Chapter`, `@Section`, and `@SubSection` command in the source is replaced by a Symbolics Concordia `Chapter`, `Section` or `SubSection` command. These commands create entries in the Table of Contents, just as records do. In this way a document can be printed even though it is not yet ready for good, modular online use.

4. If the document does not have `@` style markup commands, replace the existing formatter commands with the appropriate Symbolics Concordia commands or add the appropriate sectioning commands and other formatting commands. See the section "Concordia Markup Language", page 119, for appearance formatting commands. For sectioning commands, see the section "Sectioning Commands for Conversion Compatibility", page 429.
5. Write and evaluate a `sage::register-book` form (see the function `sage::register-book`, page 428) form for the document.
6. Use Format Pages in the Page Previewer to produce the document.



## 42. Converting Documents From Concordia to Scribe

Symbolics Concordia can output Scribe .mss files. This is particularly useful if you need to submit a paper, for example, to someone who wants it in machine readable format but does not have Symbolics Concordia.

To create a Scribe .mss file, you go to a Lisp Listener and use the Command Processor command `Convert Topic To Scribe`.

### Convert Topic to Scribe

`Convert Topic To Scribe` *documentation-topic* *pathname* *keywords*

Reads *documentation-topic* and writes a Scribe .mss file (*pathname*) that contains Scribe formatting directives to allow the topic to be formatted with Scribe.

*documentation-topic* The topic to convert.

*pathname* The pathname of a file to put the Scribe source in.

*keywords* :Output Destination, :Process Unconvertables

:Output Destination

{Buffer, File, Kill Ring, None, Printer, Stream, Window}  
Where to redirect the typeout done by this command. The default is the stream **\*standard-output\***.

:Process Unconvertables

{Yes, No} Whether to convert unconvertable Symbolics Concordia constructs into simple strings or signal an error. The default is No.



## **PART VII.**

### **SYMBOLICS CONCORDIA ADMINISTRATOR'S GUIDE**



## 43. Guide to Symbolics Concordia Book Design

### 43.1. Introduction to Symbolics Concordia Book Design

In Symbolics Concordia, you can:

- *Control* the appearance of a book by using the Symbolics Concordia Markup Language. The markup language is a set of predefined specifications for formatting environments and commands.
- *Modify* the appearance of a book by modifying these specifications. This book describes how to modify these specifications.

This section assumes you are familiar with the Symbolics Concordia Markup Language. If you are not, see the section "Controlling Your Document's Appearance", page 113.

#### 43.1.1. What is Book Design?

Simply stated, *book design* is what determines the appearance of a book. Book design utilizes typography and page layout to produce two-dimensional compositions which are pleasing to the eye, and appropriate to the subject matter and audience. You could say that book design *is* the appearance of a book.

The book design capabilities of Symbolics Concordia are very strong but, at this time, the interface to them is a Lisp-like language. See the section "How the Book Design Sources Are Organized", page 362. Some of the things described in this document are going to seem like nuts-and-bolts internals. This is because they are.

#### 43.1.2. Book Design Vocabulary

Discussions about book design can be confusing because too often terms are used interchangeably or colloquially. The following is a glossary of book design terms as used in Symbolics Concordia.

Book (Document)	A registered record. That is, a record for which a <b>sage::register-book</b> form has been evaluated. To see how to register a record, see the function <b>sage::register-book</b> , page 428.
Book Design	A set of aesthetic decisions which affect the appearance of a book or set of books.
Specification	In a general sense, a description of how something in a book should look. A specification consists of a set of definitions which control the appearance of a <i>book design element</i> . In Symbolics Concordia, specifications live in the <i>book design sources</i> as clauses in <b>sage:define-book-design</b> forms.

### Top-level Specification

Specifications in top-level style or text clauses in *top-level book design elements*. Top-level style specifications define the block in which type is formatted, that is the margins, and consists of attributes such as `paper-width`, `leftmargin`, `rightmargin`, `top-margin`, and `bottommargin`. Top-level text specifications control the appearance of running text not inside some environment and consists of attributes such as `Font`, `LineWidth`, `Spacing`, `Spread`, `FaceCode`, `Indent`, `Justification`, and `Hyphenation`.

### Book Design Element

Any aspect of a book which can be modified to change the appearance of the book. Examples of book design elements are footnotes, chapter headings, pageheadings, enumerated lists, and the like.

In Symbolics Concordia, a book design element is a collection of one or more *specifications*. Book design elements provide a way to organize all the many specifications which make up the appearance of your documentation. See the section "Book Design Elements", page 366.

### Top-level Book Design Element

A book design element whose name includes the name of a *document device type* is a top-level book design element. Such a book design element includes the top-level specifications for that particular document device type. For instance, **sage::letter-1gp2** contains the top-level specifications for the Letter LGP2 document device type.

### Document Type

A collection of *book design elements* for one particular type of book but any *device type*. "Reference", "Letter", and "Article" are document types. See the section "Document Device Types", page 363.

### Device Type

A device on which formatted documentation is output. Currently, Symbolics Concordia supports the device types Generic, Screen, LGP2, and Dex. The Generic device type does not actually represent an output device supported by Symbolics Concordia but instead a universal device type used by the other two. For more explanation of the other device types, see the section "Document Device Types", page 363.

### Document Device Type

A collection of *book design elements* which together specify the appearance of formatted documentation. A document device type is made up of a *document type* and *device type*. "Reference LGP2", "Generic Screen", and "Lisp-dictionary LGP2" are document device types. See the section "Document Device Types", page 363.



## Book Design Sources

A set of Lisp code files which contain all the book design specifications for Symbolics Concordia. See the section "How the Book Design Sources Are Organized", page 362.

Here are some additional terms used in Symbolics Book Design:

Collectors	Specifications for the table of contents, table of figures, table of tables, and index. Collectors collect entries as the book is formatted. For instance, the contents collector collects section headings and their page numbers as the book is being formatted. Then, when it is time to format the table of contents, the contents collector supplies the necessary information. See the function <b>sage:define-book-design</b> , page 421.
Command	A directive to the formatter. See the section "Markup Commands and How to Use Them", page 118.
Counter	A specification which defines how a chapter, section, or other section heading gets formatted. See the function <b>sage:define-book-design</b> , page 421.
Environment	A collection of directives to the formatter that are bundled together (usually with some high level intent) that take effect over a region (defined by the environment markup delimiters) of a document. See the section "Environments and How to Use Them", page 113.

### 43.1.3. Inheritance Dependency

Book design elements can inherit from other book design elements and environments can inherit from other environments. Children inherit from their parents and grandparents. Therefore, modifying something about a parent might affect its children and its grandchildren. You must consider these *inheritance dependencies* carefully when deciding where to make modifications to book design elements and environments. The "Book Design Browser" provides tools for researching these inheritance dependencies. See the section "Book Design Browser", page 369. See the section "Modifying a Document Device Type", page 373.

In the book design sources, a "use" clause introduces a parent book design element or environment. The book design element or environment which includes a "use" clause is the child. For example, consider the following book design element:

```
(define-book-design child-book-design ()
  (use parent-book-design)
  (define
    (child-envr (use parent-envr))))
```

Several "Book Design Browser" commands provide you with information about inheritance dependencies.

**Describe Document Device Type**

Lists the given document device type's top-level book design element and that element's parents.

**Describe Book Design Element**

Lists the given book design element's parents.

**Show Dependencies on Book Design Element**

Lists the given book design element's children.

**Describe Environment**

Displays a table of the given environment's parent environments and its parent book design elements and lists the given environment's children.

**Show Dependencies on Environment**

Lists the given environment's children.

In the displays produced by these commands:

- "Inherits from" or "is specified by" introduces a list of *parents*, *grandparents*, and so on. In this example, **sage::generic-generic** (child) inherits from **sage::page-heading-environments** (parent), and so on. Note that indented elements are next-generation parents. So, in this example, **sage::user-visible-environments** (child) inherits from **sage::inner-environments** (parent).

**GENERIC-GENERIC** is a book design element.

```
Inherits from:
PAGE-HEADING-ENVIRONMENTS
SECTION-HEADING-ENVIRONMENTS
TABLE-OF-CONTENTS-ENVIRONMENTS
NUMBERED-SECTIONS
OTHER-COUNTERS
STANDARD-COLLECTORS
USER-VISIBLE-ENVIRONMENTS
  INNER-ENVIRONMENTS
FACECODE-ENVIRONMENTS
DEFLINE
```

- "Is inherited by" introduces a list of *children*.
  - For a book design element, "is inherited by" introduces a list of *children*, *grandchildren*, and so on. In this example, **sage::inner-environments** (parent) is inherited by **sage::user-visible-environments** (child), and so on. Note that indented elements are next generation children. So, in this example, **sage::user-visible-environments** (parent) is inherited by **sage::generic-generic** (child).

**INNER-ENVIRONMENTS** (a book design element) is inherited by:

```

USER-VISIBLE-ENVIRONMENTS
  GENERIC-GENERIC
    GENERIC-SCREEN
      GENERIC-DEX
        GENERIC-DEX-BACKGROUND
      MEMO-SCREEN
      LETTER-SCREEN
      GENERIC-MAC-SCREEN
    GENERIC-PAPER
      GENERIC-LGP2
        3SYMANUAL-LGP2
          LISP-DICTIONARY-LGP2
          ARTICLE-LGP2
          DOC-EX-LGP2
          MAC-DOC-LGP2
        APPROACH-LGP2
        MEMO-LGP2
        LETTER-LGP2
        INSTALLATION-LGP2
      GENERIC-DMP1
    MEMO-GENERIC
      MEMO-LGP2
      MEMO-SCREEN
    LETTER-GENERIC
      LETTER-SCREEN
      LETTER-LGP2
    ARTICLE-GENERIC

```

- For an environment, "is inherited by" introduces a list of *children*.  
**Description** (an environment) is inherited by:

<u>Environment</u>	<u>In Book Design Element</u>
Transparent	INSTALLATION-ENVR-MODS
Descript	APPROACH-ENVR-MODS
Description2	TOOLS-AND-TECHNIQUES-ENVIRONMENTS
Descript	USER-VISIBLE-ENVIRONMENTS

#### 43.1.4. Modifying the Appearance of Your Documentation

You can modify the appearance of your documentation at any of three levels. You must first decide which level is appropriate.

To modify the appearance of:

- All books of a particular document device type, you modify the book design sources for that particular document device type. See the section "How the Book Design Sources Are Organized", page 362. See the section "Modifying a Document Device Type", page 373..
- A particular book, use the Symbolics Concordia command "Modify" in the top-level record of that book. See the section "Modifying a Single Book", page 384.

- A single instance of a markup environment, modify its attributes by clicking Right on one of its markup delimiters. See the section "Modifying a Single Instance of an Environment", page 389.

## 43.2. How the Book Design Sources Are Organized

When you want to modify a document device type, you must modify the book design sources for that document device type. The book design sources contain the design specifications which control the appearance of the various document device types. The sources are Lisp code files in the directory `SYS:NSAGE;`. The code used in these files is written in Lisp.

The "Book Design Browser" provides tools to help you investigate the contents of these files. It saves you from having to wade through the code in them and it assists you in identifying where to make modifications. See the section "Book Design Browser", page 369.

See the section "Modifying a Document Device Type", page 373 for examples of how to use the "Book Design Browser" to make modifications to the book design sources.

The sources include definitions for document types, book design elements, commands, environments, attributes, counters, fonts, and much more. Mostly what you will be concerned with are "Document Device Types" and "Book Design Elements".

### 43.2.1. Book Design Source Files

The following is a listing of the relevant files in the `SYS:NSAGE;` directory with a description of the contents of each.

<code>BD-APPROACH.LISP</code>	The book design specifications for the Approach LGP2 document device type.
<code>BD-ARTICLE.LISP</code>	The book design specifications for the Article Generic and Article LGP2 document device types. This file also defines several environments used just by these document device types.
<code>BD-DICTIONARY.LISP</code>	The book design specifications for the Lisp-Dictionary LGP2 document device type. This file also defines several Lisp functions and macros used to format Lisp dictionaries.
<code>BD-GENERIC.LISP</code>	The top-level book design elements for the various Generic document types. The book design specifications for these document types are defined in <code>database.lisp</code> .
<code>BD-INSTALLATION.LISP</code>	The book design specifications for the Installation LGP2 document device type.
<code>BD-LETTER.LISP</code>	The book design specifications for the Letter Generic, Letter Screen, and Letter LGP2 document device types. Various commands and environments used only by the Letter document types are defined in <code>bd-letter-commands.lisp</code> .

BD-LETTER-COMMANDS.LISP	Various commands and environments used only by the Letter document types.
BD-MEMO.LISP	The book design specifications for the Memo Generic, Memo Screen, and Memo LGP2 document device types. Various commands and environments used only by the Memo document types are defined in BD-LETTER-COMMANDS.LISP.
BD-MEMO-COMMANDS.LISP	Various commands and environments used only by the Memo document types.
BD-REFERENCE-CARDS.LISP	The book design specifications for the Reference LGP2 document device type.
DATABASE.LISP	The book design specifications for the 3symanual LGP3 document device type. In addition, all the various environments, attributes, counters, collectors, commands, fonts, and boxes are defined here.

### 43.2.2. Document Device Types

A book is formatted according to the book design specifications of two components: a *document type* and a *device type*.

- A document type supplies the specifications for the kind of a document you are formatting, for instance a letter or reference cards.
- A device type supplies the characteristics for the output device you intend to use, the screen or a Symbolics LGP2, for instance.
- A document device type is the combination of these two components, for instance Letter Screen or Reference LGP2.

Document device types are defined by the following Lisp functions and macros:

Device type           **sage::define-device-type.**  
 Document type       **sage::define-document-type.**  
 Document device type           **sage:define-book-design** and **sage:note-book-design-specifics.**

So, for instance, the following code defines the Reference Cards document type for the LGP2 output device.

```
(define-device-type 'lgp2)

(define-document-type 'reference-cards)
```

```

(define-book-design reference-cards-lgp2 ()
  (use generic-lgp2)
  (use approach-lgp2)
  (use reference-cards-counters)
  (use reference-cards-headings)
  (use reference-cards-envr-mods)
  (modify
    (box (BoxFlushRight Yes))
  ;; (contentsenv (rightmargin "+2inches"))
    (bodystyle (spacing 11pts) (spread 2pts))
    (K (FACECODE K) TABEXPORT (HYPHENBREAK off))
  )
  (first
    ;;Style
    (envr (text
      (leftmargin "2.0inches")
      (RightMargin "2.0inches")
      (BottomMargin "3.0inches")
      (TopMargin "1.0inches")))
    ;;Top-level begin
    (envr (Text
      (Font SmallBodyFont)
      ;;(leftmargin "2.25inches")
      (LineWidth 26picas)
      (Spacing 11pts)
      (Spread 2pts)
      (FaceCode R)
      (Indent 0)
      (Justification on)
      (Hyphenation On)
      (Spaces Compact)))
    (init
      initialize-reference-cards-lgp2)
  ))

(note-book-design-specifics 'reference-cards :lgp2 'reference-cards-lgp2)

```

When you format a particular book, you specify the document type by using the `:document-type` keyword to **sage::register-book**. If you want the book to be an article, for instance, then you would write the **sage::register-book** form as follows:

```

(sage::register-book "A Record Name"
  :document-type 'sage::article)

```

When you format a particular book, you specify the device type as follows:

<i>To use device type</i>	<i>Format record using</i>
LGP2	Page Previewer command Format Pages Command Processor command Show Documentation :destination keyword Editor command c-U m-X Show Documentation Editor command c-U s-P
Screen	Command Processor command Show Documentation m-X Show Documentation s-P
Dex	Document Examiner command Show Documentation

The following matrix shows the document device type used for particular formatting conditions:

	<i>registered book</i>	<i>record from registered book</i>	<i>non-registered record</i>
<i>Format Pages</i>	<i>document-type lgp2</i>	<i>document-type lgp2</i>	<i>generic lgp2</i>
<i>Show Documentation</i>	<i>generic screen</i>	<i>generic screen</i>	<i>generic screen</i>
<i>m-X Show Documentation</i>	<i>generic screen</i>	<i>generic screen</i>	<i>generic screen</i>
<i>s-P</i>	<i>generic screen</i>	<i>generic screen</i>	<i>generic screen</i>
<i>c-U Show Documentation</i>	<i>generic paper</i>	<i>generic paper</i>	<i>generic lgp2</i>
<i>c-U s-P</i>	<i>generic paper</i>	<i>generic paper</i>	<i>generic lgp2</i>

Note: *document-type* means whatever value you've given for **:document-type** in the **sage::register-book** form for that book.

The following matrix describes the current set of document device types provided in Symbolics Concordia Book Design:

<i>device type</i> <i>document type</i>	generic	screen	lgp2
3symanual			3symanual lgp2
approach			approach lgp2
article	article generic		article lgp2
generic	generic generic		generic lgp2
installation			installation lgp2
letter	letter generic	letter screen	letter lgp2
memo	memo generic	memo screen	memo lgp2
lisp-dictionary			lisp-dictionary lgp2
reference-cards			reference-cards lgp2

### 43.2.3. Book Design Elements

A book design element is a collection of one or more specifications. In the book design sources, book design elements are defined using the function **sage:define-book-design**. See the function **sage:define-book-design**, page 421.

The name of a book design element indicates what kind of specifications it contains. For instance, **sage::section-heading-environments** is a book design element which contains the specifications for environments used in formatting section headings.

A book design element whose name includes the name of a document device type is a top-level book design element. Such a book design element includes the top-level specifications for that particular document device type. For instance, **sage::letter-lgp2** contains the top-level specifications for the Letter LGP2 document device type.

The command "Describe Book Design Element" displays information about a given book design element. The command indicates what sort of specifications the book design element includes. The command also lists any other book design elements inherited by the given element. For top-level book design elements, the display also includes the top-level specifications. See the section "Describe Book Design Element", page 412.

You can modify the appearance of a document device type by modifying the book design elements which specify that document device type. For more information on modifying book design elements, see the section "Modifying a Document Device Type", page 373.

The following is a listing of the various book design elements for the generic document type. Some of these, as noted, are simply placeholders and not inherited by



any other book design elements. The others are simply collections of book design elements which are inherited by other book design elements. They serve to modularize the book design sources.

**sage::generic-dex** A placeholder. Not intended for use at this time.

**sage::generic-dex-background**

A placeholder. Not intended for use at this time.

**sage::generic-dmp1**A placeholder. Not intended for use at this time.

**sage::generic-generic**

A collection of book design elements inherited by several other book design elements.

**sage::generic-lgp2** A collection of book design elements inherited by several other book design elements.

**sage::generic-mac-screen**

A placeholder. Not intended for use at this time.

**sage::generic-paper**

A collection of book design elements inherited by several other book design elements.

**sage::generic-screen**

A collection of book design elements inherited by several other book design elements.

The following is a description of the currently defined *top-level* book design elements. These are the book design elements which contain top-level specifications for the various document device types.

**sage::3symanual-lgp2**

The top-level specifications for the 3symanual LGP2 document device type.

**sage::approach-lgp2**

The top-level specifications for the Approach LGP2 document device type.

**sage::article-generic**

The top-level specifications for the Article Generic document device type.

**sage::article-lgp2** The top-level specifications for the Article LGP2 document device type.

**sage::installation-lgp2**

The top-level specifications for the Installation LGP2 document device type.

**sage::letter-generic**A top-level specification for the Letter Screen and Letter LGP2 document device types.

**sage::letter-*lgp2*** Top-level specifications for the Letter L<sub>GP2</sub> document device type.

**sage::letter-*screen*** Top-level specifications for the Letter Screen document device type.

**sage::lisp-*dictionary-lgp2***  
Top-level specifications for the Lisp Dictionary L<sub>GP2</sub> document device type.

**sage::memo-*generic***  
Top-level specifications for the Memo document types.

**sage::memo-*lgp2*** Top-level specifications for the Memo L<sub>GP2</sub> document device type.

**sage::memo-*screen*** Top-level specifications for the Memo Screen document device type.

The following is a listing of all the rest of the book design elements.

**sage::*approach-counters***  
The specifications for section counters for the Approach document type.

**sage::*approach-envr-mods***  
Modifications to environments for the Approach document type.

**sage::*approach-headings***  
Modifications to environments used in section headings for the Approach document type.

**sage::*article-counters***  
The specifications for section counters for the Article document type.

**sage::*article-headings***  
Modifications to environments used in section headings for the Article document type.

**sage::*defline*** The specifications for environments used in the first line of a Lisp object record.

**sage::*dmp1-inner-styles***  
A placeholder. Not intended for use at this time.

**sage::*facecode-environments***  
The specifications for the facecode environments, for instance B (for boldfaced type) and I (for italicized type).

**sage::*inner-environments***  
Specifications for environments which are not intended for use in your documentation. Instead, these environments are inherited by other environments.

**sage::lgp2-inner-styles**

Specifications for environments used in several document types for the LGP2 device type.

**sage::lisp-dictionary-counters**

Specifications for counters used in the Lisp Dictionary document type.

**sage::lm-inner-styles**

A placeholder. Not intended for use at this time.

**sage::numbered-sections**

Specifications for the numbered section counters.

**sage::other-counters**

Specifications for FigureCounter, TableCounter, and FootNoteCounter counters.

**sage::page-heading-environments**

Specifications for environments used in pageheadings and pagefootings.

**sage::screen-inner-styles**

Specifications for environments used in various document types for the Screen device type.

**sage::section-heading-environments**

Specifications for environments used in section headings.

**sage::standard-collectors**

Collectors for the table of contents, table of figures, table of tables, and index.

**sage::table-of-contents-environments**

Specifications for environments used in the table of contents, table of figures, and table of tables.

**sage::tools-and-techniques-environments**

Specifications for some environments not currently used anywhere else.

**sage::user-visible-environments**

Specifications for environments available for general use.

## 43.3. Book Design Browser

### 43.3.1. Introduction to the Book Design Browser

The Symbolics Concordia Book Design Browser is a book design maintenance and development tool. For definitions of the terms used in the Browser, see the section "Book Design Vocabulary", page 357.

You can use the Browser to:

- Investigate the default definitions of Concordia environments for particular document device types.
- Change those default definitions.
- Develop new document device types for your site.

Select the Browser by clicking on the Browser Icon in the upper right-hand corner of the Symbolics Concordia window.



Figure 51. Book Design Browser Icon

Once you've selected the Book Design Browser, click on Help in the menu pane to display some introductory documentation about the Book Design Browser.

For a description of the Book Design Browser, see the section "Layout of the Book Design Browser Window", page 370.

### 43.3.2. Layout of the Book Design Browser Window

Figure 52 shows the initial Book Design Browser.

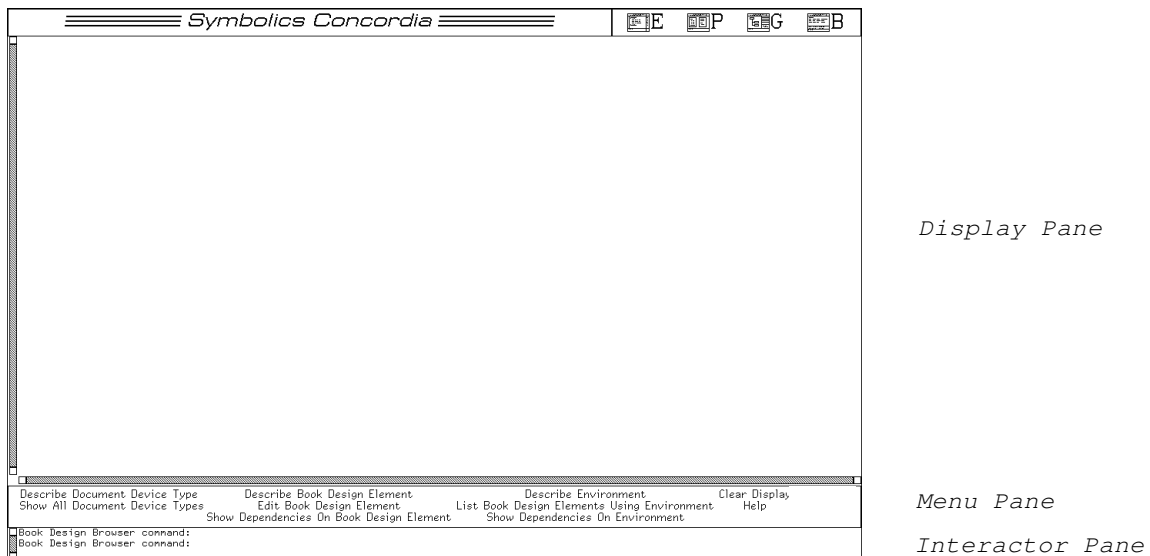


Figure 52. Initial Book Design Browser

The Book Design Browser consists of the "Symbolics Concordia" logo in the upper left of the pane; the icon pane in the upper right-hand corner of the pane; and the following additional panes:

<i>Pane</i>	<i>Description</i>
-------------	--------------------

Display Pane	Information produced by the Book Design Browser commands is displayed in this pane. Much of the information displayed here is mouse sensitive, meaning that you can perform many operations on the information by using the mouse. See the section "Mouse Sensitivity in the Book Design Browser", page 373.
Menu Pane	The Book Design Browser commands are all listed here and each can be run by clicking Left on its name. When you click on a command name, the command appears in the Interactor Pane and prompts you for any information it might need. Note that clicking Middle on a command name displays the documentation about that command. Clicking on Help in the menu pane displays some introductory documentation about the Book Design Browser.
Interactor Pane	You can type the Book Design Browser commands here. A command prompts you for any information it might need. When you click on a command name in the Menu Pane, the command appears here and prompts you for the information it needs to run.

To clear the display pane use the command "Clear Display". Information displayed in the display pane is never actually killed. You can scroll backward and forward using the scroll bar to the left.

### 43.3.3. Getting Help in the Book Design Browser

To see documentation for the commands, do one of the following:

- Click Middle on the command name in the menu.
- Use the Show Documentation command as follows:

Show Documentation *command-name*

To see documentation for environments and their attributes listed in displays, do one of the following:

- Click Middle on the environment or attribute name in a display. (Note that not all environments and attributes are documented. See the mouse documentation line while pointing at an environment or attribute name to make sure.)
- Use the Show Documentation command as follows:

Show Documentation *environment* Environment

or

Show Documentation *attribute* Attribute

#### 43.3.4. Issuing Commands in the Book Design Browser

To issue a command in the Book Design Browser, do one of the following:

- Type the command name at the "Book Design Browser command:" prompt in the Interactor Pane.
- Click Left or Right on the command name in the Menu Pane.
- Click Right on a mouse-sensitive item in a Display Pane display and choose a command from the pop-up menu.

If you're not sure what the allowable arguments for a particular command are, do one of the following:

- Check the documentation for that command by clicking Middle on the command name in the Menu Pane.
- To see a listing of possibilities when a command prompts you for an argument, press the HELP key. For instance:

```
Describe Document Device Type (document device type) <HELP>
=>
You are being asked to enter a document device type.
```

```
These are the possible document device types:
3SYMANUAL LGP2   GENERIC DEX           GENERIC LGP2 ...
APPROACH LGP2   GENERIC DEX-BACKGROUND  GENERIC MAC-SCREEN ...
ARTICLE GENERIC  GENERIC DMP1           GENERIC SCREEN ...
ARTICLE LGP2    GENERIC GENERIC         LETTER GENERIC ...
```

Except for "Clear Display" and "Edit Book Design Element", each of the Book Design Browser commands takes an argument which controls where the command's output is sent. The choices are:

Display	Output is displayed in the Display Pane of the Browser. This is the default.
Printer	Output is printed on your default printer. (Note that the output from "Describe Environment" is printed in landscape format since it can be very wide.)
Typeout	Output is displayed in a temporary typeout window. This is the same as if you clicked Right on a mouse-sensitive item in the Display Pane, and then on one of the commands in the pop-up menu.

Here is an example which directs its output to your default printer:

Describe Environment (environment) Example (a document device type) GENERIC GENERIC (where [default Display]) Printer

For a descriptions of Book Design Browser commands, see the section "Dictionary of Book Design Browser Commands", page 412.

### 43.3.5. Mouse Sensitivity in the Book Design Browser

Many items in Book Design Browser displays are mouse sensitive. Mouse sensitivity is indicated by a box outlining the item when you point the mouse at it. While pointing at a mouse-sensitive item, note the mouse documentation line for actions you can take on that item. "Mouse R: Menu" means that a menu of more commands pops up when you click Right on that item. For instance, you can see documentation for some attributes and environments by clicking Right on a mouse-sensitive attribute or environment name, and then click on "Show Documentation" in the pop-up menu.

## 43.4. Modifying a Document Device Type

### 43.4.1. Introduction

This section discusses how to modify the appearance of your documentation at the document device type level. Modifications made at this level affect all books formatted using that particular document device type. "Book Design Browser" is a Symbolics Concordia tool which facilitates this work. See the section "Book Design Browser", page 369.

You can modify two kinds of specifications at the document device type level:

- You can modify specifications for environments. When you modify an environment at the document device type level, the modification will appear in all instances of that environment in any book formatted using that document device type. See the section "An Example of a Simple Modification to an Environment in a Document Device Type", page 374. See the section "An Example of a Modification to an Environment with Dependencies in a Document Device Type", page 378.
- You can modify top-level specifications for a document device type. See the section "An Example of a Modification to a Top-level Specification in a Document Device Type", page 380.

When you make modifications at the document device type level, it is essential that you consider issues of inheritance dependencies. Remember that book design elements can inherit from other book design elements and environments can inherit from other environments. Such inheritance can create dependencies which you must keep in mind in making modifications at the document device type level. The "Book Design Browser" provides you with tools to see and identify these inheritance dependencies. See the section "Inheritance Dependency in Symbolics Concordia Book Design", page 359.

Put as simply as possible, to modify the appearance of your documentation at the document device type level you must:

1. Decide what you want to modify and how you want it to look.
2. Identify the current specification for the thing you want to modify.
3. Investigate any inheritance dependancies and thereby decide where to make your modification.
4. Make the modification.

To begin, select the Book Design Browser by clicking on Book Design Browser icon.



Figure 53. Book Design Browser Icon

#### 43.4.2. Examples of Document Device Type Modifications

Some knowledge about book design elements, environments, counters, and attributes is assumed in these example. To find out about a particular book design element, use the command "Describe Book Design Element". To find out about a particular environment, counter, or attribute, see its documentation. See the section "Getting Help in the Book Design Browser", page 371.

##### 43.4.2.1. An Example of a Simple Modification

This example makes a relatively simple modification to an environment for all document device types. There are no issues of inheritance dependency raised in this example. For an example of a modification with inheritance dependency issues, see the section "An Example of a Modification to an Environment with Dependencies in a Document Device Type", page 378.

1. **Decide what you want to modify and how you want it to look.**

Let's say you want to modify the appearance of the "Itemize Environment" and you want that modification to appear in all document device types. You want the items in this environment to format without any blanklines between them. The attribute you need to modify is the "Spread Attribute". To affect all document device types, you need to modify Itemize in some book design element which is inherited by all other document device types. We can guess that this will be the Generic Generic document device type since it is inherited by all other document device types.



## 2. Identify the current specification.

See what the current specification for Itemize in Generic Generic is by using the "Describe Environment" command as follows:

```
Describe Environment (environment) Itemize (a document device type)
GENERIC GENERIC
```

This command produces the display in figure 54, page 375.

**Itemize** (an environment) is defined in **GENERIC GENERIC** (a document device type) as follows:

Attribute	Value	From Environment	In Book Design Element	Inherited by Book Design Element
Break		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Continue		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Fill		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
LeftMargin	+2	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Indent	-2	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
RightMargin	+0	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Numbered	@@s(o) @,@@s(.)	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
NumberLocation	LFR	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
BlankLines	BREAK	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Spacing	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Above	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Below	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Spread	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
ParagraphBreaks	NORMAL	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC

**Itemize** (an environment) is inherited by:

Environment	In Book Design Element
Enumerate	APPROACH-ENVR-MODS
Checklist	USER-VISIBLE-ENVIRONMENTS
Enumerate	USER-VISIBLE-ENVIRONMENTS
Checklist	3SYMANUAL-LGP2

Figure 54. Describe Environment Itemize GENERIC GENERIC

In the first column, under "Attribute", look for "Spread". Notice its current value, in the second column under "Value", is "1". This means "leave one blank line between each item in an itemized list". You want to modify this to be "0" which means "don't leave any blanklines between items".

## 3. Investigate any inheritance dependencies and decide where to make your modification.

- a. In the third column, under "From Environment", you can see that this attribute and its value come from the definition of Itemize itself. This is helpful because if this attribute and its value were inherited from some other environment, then you would have to consider the inheritance dependency consequences of making the modification. Since this attribute and its value come from the definition of Itemize itself, it is safe to proceed.
- b. In the fourth column, under "In Book Design Element", notice that the definition of Itemize occurs in **safe::user-visible-environments**. Since the name of the book design element is in **boldface**, you know that this is where Itemize is defined. If the name of the book design element is in

*italics*, then you know that the environment is modified in that book design element.

In the last column, under "Inherited by Book Design Element", you can see that **sage::generic-generic** is the book design element which inherits **sage::user-visible-environments**. This is exactly what you want because you know that **sage::generic-generic** is the top-level book design element for the Generic Generic document device type. What this all tells you is that any modification to Itemize in **sage::user-visible-environments** is inherited by any book design element which inherits **sage::user-visible-environments**, for instance **sage::generic-generic**. You can see the dependencies on these book design elements by clicking Right on "user-visible-environments" and then on "Show Dependencies on Book Design Element".

These are important pieces of information because they can indicate inheritance dependencies which you might need to consider in making your modification. For more discussion of these considerations, see the section "An Example of a Modification to an Environment with Dependencies in a Document Device Type", page 378.

- c. Next, you need to check for any inheritance dependencies on Itemize itself. If any other environments inherit Itemize, they are listed at the bottom of the display. In this case, Itemize is inherited by several other environments in several other book design elements. This list tells you that Enumerate and Checklist inherit Itemize in their definitions. Therefore, by modifying Itemize, you are going to affect Enumerate and Checklist. For discussion's sake, let's assume that this is OK and therefore it is safe to proceed.

#### 4. Make the modification.

- a. Now that you have investigated any and all inheritance dependencies, you can make your modification. Since the Spread attribute in Itemize is set in the **sage::user-visible-environments** book design element, and you have concluded that that is the appropriate place to make this modification, click Right on "user-visible-environments" and then on "Edit Book Design Element" in the pop-up menu. The Zmacs editor is selected, the source file containing the **sage::user-visible-environments** book design element is read into an editor buffer (if it was not already) and selected, and the cursor is positioned at the beginning of the definition of **sage::user-visible-environments**. Find the definition of Itemize within **sage::user-visible-environments**. (If the definition is not immediately visible, you can search for it using the Incremental Search command `c-S`.)

The relevant sections from the source for the book design element **sage::user-visible-environments** looks like this. (The "→→" characters are added here to show you where you want to make you modification.)

```

(define-book-design user-visible-environments ()
  (define
    (Itemize Break
      Continue
      Fill
      (Leftmargin "+2")
      (Indent "-2")
      (Rightmargin "+0")
      (Numbered "@@S(o) @,@@S(.) ")
      (Numberlocation Lfr)
      (Blanklines Break)
      (Spacing 1)
      (Above 1)
      (Below 1)
      →→ (Spread 1)
      (Paragraphbreaks normal))))

```

- b. Change the value of the Spread attribute from 1 to 0.
- c. Compile the change by using the command `c-sh-C`. This adds the change you've made to your machine's memory. (Note that unless you patch this modification into a distributed system, you will lose the modification next time you boot your computer. See the following step for more information about this.) Now, when you format an Itemize environment in any document device type, no blank lines will be left between items. Now, the display produced by Describe Environment looks like figure 55, page 377.

**Itemize** (an environment) is defined in **GENERIC GENERIC** (a document device type) as follows:

Attribute	Value	From Environment	In Book Design Element	Inherited by Book Design Element
Break		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Continue		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Fill		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
LeftMargin	+2	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Indent	-2	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
RightMargin	+0	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Numbered	@@S(o) @,@@S(.)	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
NumberLocation	LFR	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
BlankLines	BREAK	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Spacing	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Above	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Below	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Spread	0	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
ParagraphBreaks	NORMAL	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC

**Itemize** (an environment) is inherited by:

Environment	In Book Design Element
Enumerate	APPROACH-ENVR-MODS
Checklist	USER-VISIBLE-ENVIRONMENTS
Enumerate	USER-VISIBLE-ENVIRONMENTS
Checklist	3SYMANUAL-LGP2

Figure 55. Itemize for Generic Generic Document Device Type with the Spread Attribute Modified.

- d. At this point, the modification you've made is known only to your computer. If you format a document on another computer, you will not see your modification. Likewise, if you boot your computer, your modification will be lost. To have the modification available even if you boot your computer, and to distribute your modification to other users at your site, you should patch the modification into a documentation system. For instructions on how to do this, see the section "Defining a Book Design System for a Site", page 410, and see the section "Patching a Documentation System", page 442.

#### 43.4.2.2. An Example of a Modification to an Environment with Dependencies

1. **Decide what you want to modify and how you want it to appear.**

Let's say you want to modify Enumerate for all document device type so that no blank lines appear between items in the enumerated list. The attribute you need to modify is "Spread Attribute". Since you want this modification to affect all document device types, you have to make the modification someplace that is used by all other document device types. We know that the Generic Generic document device type is used by all other document device types, so that's a good place to start.

2. **See what the current definition of Enumerate is for the Generic Generic document device type.** Use the following command:

```
Describe Environment (environment) Enumerate (a document device type)
GENERIC GENERIC
```

This command produces the display in figure 56, page 378.

**Enumerate** (an environment) is defined in **GENERIC GENERIC** (a document device type) as follows:

Attribute	Value	From Environment	In Book Design Element	Inherited by Book Design Element
LeftMargin	+4	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Indent	-4	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Numbered	@1. @, @a. @, @i.	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Referenced	@1@, @a@, @i	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Break		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Continue		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Fill		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
RightMargin	+0	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
NumberLocation	LFR	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
BlankLines	BREAK	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Spacing	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Above	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Below	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Spread	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
ParagraphBreaks	NORMAL	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC

**Enumerate** (an environment) is not inherited by any other environment.

Figure 56. Describe Environment Enumerate GENERIC GENERIC

In the first column, under "Attribute", look for the attribute "Spread". Notice its current value, in column two under "Value", is "1" which means "leave one blank line between each item in an enumerated list". You want to change this to "0" which means "don't leave any blank lines between items".

### 3. Investigate any inheritance dependancies and decide where to make your modification.

- a. In the third column, under "From Environment", notice that this attribute and its value are inherited from Itemize. This means that the Spread attribute is set inside the definition of Itemize and not inside the definition of Enumerate. But you want to modify Enumerate not Itemize. What this tells you is that instead of *making* a modification to an attribute's value, you must *add* a modification to the definition of Enumerate.
- b. Look for where Enumerate is defined. In the fourth column, under "In Book Design Element", you can see that Enumerate is defined in **sage::user-visible-environments** because that book design element name is in **boldface** next to "Enumerate". This is where you can add your modification.

### 4. Make your modification.

- a. Click Right on **sage::user-visible-environments** and then on "Edit Book Design Element" in the pop-up menu. The Zmacs editor is selected, the source file containing the **sage::user-visible-environments** book design element is read into a buffer (if it was not already) and selected, and the cursor is positioned at the beginning of **sage::user-visible-environments**. Find the definition of Enumerate within **sage::user-visible-environments**. (If the definition is not immediately visible, you can search for it using the Incremental Search command c-S.)

The relevant sections from the source for the book design element user-visible-environments looks like this.

```
(define-book-design user-visible-environments ()
  (define
    (Enumerate (Use Itemize)
      (Leftmargin "+4")
      (Indent "-4")
      (Numbered "@1. @,@a. @,@i. ")
      (Referenced "@1@,@a@,@i"))))
```

Inside the definition of Enumerate, add "Spread 0" as follows:

```
(define-book-design user-visible-environments ()
  (define
    (Enumerate (Use Itemize)
      (Spread 0)
      (Leftmargin "+4")
      (Indent "-4")
      (Numbered "@1. @, @a. @, @i. ")
      (Referenced "@1@, @a@, @i"))))
```

- b. Compile the change by using the command `c-sh-c`. This adds the change you've made to your machine's memory. Now, when you format an Enumerate environment in any document device type, no blank lines will be left between items. Now, the display produced by Describe Environment looks like figure 57, page 380.

**Enumerate** (an environment) is defined in **GENERIC GENERIC** (a document device type) as follows:

Attribute	Value	From Environment	In Book Design Element	Inherited by Book Design Element
Spread	0	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
LeftMargin	+4	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Indent	-4	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Numbered	@1. @, @a. @, @i.	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Referenced	@1@, @a@, @i	Enumerate	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Break		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Continue		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Fill		Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
RightMargin	+0	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
NumberLocation	LFR	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
BlankLines	BREAK	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Spacing	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Above	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
Below	1	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC
ParagraphBreaks	NORMAL	Itemize	<b>USER-VISIBLE-ENVIRONMENTS</b>	GENERIC-GENERIC

**Enumerate** (an environment) is not inherited by any other environment.

Figure 57. Enumerate for Generic Generic Document Device Type with the Spread Attribute Modified.

5. At this point, the modification you've made is known only to your computer. If you format a document on another computer, you will not see your modification. Likewise, if you boot your computer, your modification will be lost. To have the modification available even if you boot your computer, and to distribute your modification to other users at your site, you should patch the modification into a documentation system. For instructions on how to do this, see the section "Defining a Book Design System for a Site", page 410, and see the section "Patching a Documentation System", page 442.

### 43.4.2.3. An Example of a Modification to a Top-level Specification in a Document Device Type

In this example you can see how to make a modification to a top-level specification in a document device type. Top-level specifications are specifications in top-level style or text clauses in top-level book design elements. See the section "Book Design Elements", page 366.

Since this example concerns itself with top-level specifications for a document device type, you must start your research at the document device type level instead of at the environment level as in the other examples. Here's how to proceed:

1. **Decide what you want to modify and how you want it to appear.**

Let's say you want the top-level leftmargin to be 2 inches in the 3symanual LGP2 document device type. That's slightly larger than it currently is. This means that for any book formatted as a 3symanual LGP2 document device type, all running text (that is, all text not inside an environment, pageheading or pagefooting, or section heading, has a leftmargin of 2 inches.)

2. **Find out where the current top-level leftmargin is set in the 3symanual LGP2 document device type.**

Since top-level specifications are set in top-level book design elements, you must first find out which book design element is the top-level one for the 3symanual LGP2 document device type.

- a. Use the following command to see what book design elements make up the specification for the 3symanual LGP2 document device type:

```
Describe Document Device Type (document device type) 3SYMANUAL LGP2
```

The command produces the display in Figure 58.

**3SYMANUAL LGP2** (a document device type) is specified by:

```

3SYMANUAL-LGP2
  GENERIC-LGP2
    GENERIC-PAPER
      GENERIC-GENERIC
        PAGE-HEADING-ENVIRONMENTS
        SECTION-HEADING-ENVIRONMENTS
        TABLE-OF-CONTENTS-ENVIRONMENTS
        NUMBERED-SECTIONS
        OTHER-COUNTERS
        STANDARD-COLLECTORS
        USER-VISIBLE-ENVIRONMENTS
          INNER-ENVIRONMENTS
          FACECODE-ENVIRONMENTS
          DEFLINE
        LGP2-INNER-STYLES
      DEFLINE

```

Figure 58. Describe Document Device Type 3symanual LGP2

The display lists the family tree of book design elements for the 3symanual LGP2 document device type. You can see from the display that two of the book design elements are "top-level", those in boldface. Top-level elements set the top-level style and text specifications so one of these is

the book design element you want to look at. Since there are two listed, you must decide which of them is the appropriate place to make your modification. From its name, it is safe to assume that **sage::3symanual-lgp2** is the place to look. See the section "Book Design Elements", page 366.

- b. Use the following command to see the specification for **sage::3symanual-lgp2**:

```
Describe Book Design Element (a book design element) 3SYMANUAL-LGP2
```

The command produces the display in Figure 59.

**3SYMANUAL-LGP2** is a top-level book design element.

```
Defines or modifies some environments.
Inherits from:
  GENERIC-LGP2
  GENERIC-PAPER
    GENERIC-GENERIC
    PAGE-HEADING-ENVIRONMENTS
    SECTION-HEADING-ENVIRONMENTS
    TABLE-OF-CONTENTS-ENVIRONMENTS
    NUMBERED-SECTIONS
    OTHER-COUNTERS
    STANDARD-COLLECTORS
    USER-VISIBLE-ENVIRONMENTS
      INNER-ENVIRONMENTS
    FACECODE-ENVIRONMENTS
    DEFLINE
  LGP2-INNER-STYLES
  DEFLINE
Top-level Style:
  PAPER-WIDTH 8.5in
  LEFTMARGIN 8picas
  RIGHTMARGIN 8picas
  TOPMARGIN 8picas
  BOTTOMMARGIN 6picas
Top-level Text:
  INDENT 0
  USE BODYSTYLE
  SPACES COMPACT
  FONT BODYFONT
  FACECODE R
Init: INITIALIZE-3SYMANUAL-LGP2
```

Figure 59. Describe Book Design Element 3symanual-lgp2

From this display you can see that the current setting for the top-level leftmargin is "8picas" (which, at 6 picas to the inch, is 1 5/16").

3. Investigate any inheritance dependencies and decide where to make your modification.



You've actually already done most of this step since you decided earlier to look at **sage::3symanual-lgp2** instead of **sage::generic-paper**. But you should still make sure that any modifications you make to **sage::3symanual-lgp2** aren't inherited by other book design elements. So, use the following command to see if **sage::3symanual-lgp2** has any children:

```
Show Dependencies On Book Design Element (a book design element)
3SYMANUAL-LGP2
```

This command produces a list of four other book design elements which inherit **sage::3symanual-lgp2**:

```
LISP-DICTIONARY-LGP2
ARTICLE-LGP2
DOC-EX-LGP2
MAC-DOC-LGP2
```

Each of these is a top-level book design element for another document device type (you can tell from their names), so it is probably safe to assume that each of them has their own top-level specifications which would override any modification you make to **sage::3symanual-lgp2**. If you want to make sure, just click Right on each and then on "Describe Book Design Element" in the pop-up menu. You can see from the displays that each does indeed have their own top-level specifications so it is safe to proceed.

#### 4. Make your modification.

- a. Click Right on **sage::3symanual-lgp2** and then on "Edit Book Design Element" in the pop-up menu. The Zmacs editor is selected, the source file containing **sage::3symanual-lgp2** is read into a buffer (if it was not already) and selected, and the cursor is positioned at the beginning of **sage::3symanual-lgp2**.

The essential source code for **sage::3symanual-lgp2** follows with "`→→→`" marking the leftmargin attribute.

```

(define-book-design 3symanual-lgp2 ()
  (use generic-lgp2)
  (use defline)
  (define
    ...)
  (modify
    ...)
  (first
    (envr (text
      (Paper-Width "8.5in")
      →→→ (LeftMargin "8picas")
            (RightMargin "8picas")
            (TopMargin "8picas")
            (BottomMargin "6picas"))))
    (envr (text (Indent "0")
      (Use BodyStyle)
      (Spaces Compact)
      (Font BodyFont)
      (FaceCode R))))
  (init initialize-3symanual-lgp2)))

```

- b. Change the "8picas" to "2inches". This adds 11/16" to the leftmargin since 8 picas equals 1 5/16". (You could also change this value to "12picas" since that equals "2inches".)
  - c. Compile the change by using the command `c-sh-c`. This adds the change you've made to your machine's memory. Now when you format a record registered as a 3symanual LGP2 document device type you will get a 2" top-level leftmargin.
5. At this point, the modification you've made is known only to your computer. If you format a document on another computer, you will not see your modification. Likewise, if you boot your computer, your modification will be lost. To have the modification available even if you boot your computer, and to distribute your modification to other users at your site, you should patch the modification into a documentation system. For instructions on how to do this, see the section "Defining a Book Design System for a Site", page 410, and see the section "Patching a Documentation System", page 442.

### 43.5. Modifying a Single Book

You can modify the appearance of all occurrences of an environment in a single book by using the command markup `Modify`.

1. Insert the command markup in a sab buffer by typing `s-m Modify`. The command markup is inserted in the buffer opened so that all the attributes available for modification are listed.

2. Fill in the name of the environment you want to modify.
3. Set the attributes which you want to modify.
4. Click Right on one of the Modify markup icons and on "Change View" in the pop-up menu to close the display. Figure 60 shows a Modify markup which modifies the Description environment.

☞ Modify, Environment Description, Font smallbodyfont, Indent -15 Characters, Leftmargin +15

Figure 60. Modify Markup modifying Description environment.

By convention, you should place Modify command markups only in the script (top-level) record of your book. This way, you can more easily keep track of what's affecting the formatting.

#### 43.5.1. Specifying Page Headings and Footings

You can specify running heads for your document by placing a Pageheading command in your top level record. Create Markup Pageheading requests a pageheading command. A template pops up to allow you to insert the information to be printed.

PageHeading

Even: Yes **No**

Odd: Yes **No**

Immediate: Yes **No**

Left

Left

Center

Center

Right

Right

Line

Line

PageHeading

The fields in the template are:

Even and Odd	Whether this heading information is for an even page (lefthand or <i>verso</i> ) heading or an odd page (righthand or <i>recto</i> ) heading. You need two pageheading commands, one with <b>yes</b> for Even and the other with <b>yes</b> for Odd.
--------------	--

Immediate	Whether the headings are to take effect immediately, on the current page, or not until the second page. If you want the headings on the first page to differ from those on the second and subsequent pages, you need two pageheading commands, one with Immediate <b>yes</b> , followed by another with Immediate <b>no</b> .
Left, Center, and Right	The three text areas in the heading. You insert the text you want to appear in each place. To get the page number, insert the markup Value and specify the counter Page.
Line	An additional line of text, appearing under the left heading and possibly extending across the top of the page. If you place the command Replicate-Pattern followed by an underscore, you get a bar across the page. (Note: you insert Replicate-Pattern with <code>c-U s-M</code> , since it is an internal command.)

To insert literal spaces into a heading, for instance to specify that you want the current chapter title followed by 4 spaces and then the page, use the markup Hsp (horizontal space) which takes a horizontal distance as an argument.

For example:

```

☞value, Name chapter
☞hsp, Distance 4 Characters
☞value, Name page

```

Here is what the command templates might look like to print headings for this section:

<u>PageHeading</u>	<u>PageHeading</u>
Even: <b>Yes</b> No	Even: Yes <b>No</b>
Odd: Yes <b>No</b>	Odd: <b>Yes</b> No
Immediate: <b>Yes</b> No	Immediate: <b>Yes</b> No
<u>Left</u>	<u>Left</u>
☞ Value, Name page	☞ Value, Name sage::section
<u>Left</u>	<u>Left</u>
<u>Center</u>	<u>Center</u>
<u>Center</u>	<u>Center</u>
<u>Right</u>	<u>Right</u>
☞ Value, Name sage::chapter	☞ Value, Name page
<u>Right</u>	<u>Right</u>
<u>Line</u>	<u>Line</u>
<u>Line</u>	<u>Line</u>
<u>PageHeading</u>	<u>PageHeading</u>

Page footings are specified similarly.

Page headings and footings have no effect online.

### 43.5.2. Handling Tables of Contents and Frontmatter

Table of contents pages automatically have their page numbers printed in lower case roman numerals. The page counter is automatically reset after the table of contents, list of figures and list of tables so that the text starts on page 1 and page numbers are printed in arabic numerals.

You can add additional frontmatter, for example a Preface or Acknowledgments, simply by setting the page counter by hand in your top level record just before the include links for the Preface and/or Acknowledgments. Take into account how many pages the table of contents and lists of figures and tables take up and use the markup Set, into which you insert the markup Value, specifying the counter **sage::page** followed by the appropriate value.

☞set, Page 1

Then insert the appropriate pageheading and pagefooting commands to specify page headings and footings for this additional frontmatter. Just before the markup for Value Page, insert the markup Style. Select the parameter Pagenumber by clicking on it and give it the template "@i", which specifies lower case roman numerals.

```

|Pageheading
  Even: Yes No
  Odd: Yes No
  Immediate: Yes No
|Left
|Left
|Center
|Center
|Right
  ⌘ style, Pagenumber ((#\i))
  ⌘ value, Name page

|Right
|Line
|Line
|Pageheading

```

Then, following the links to the Preface and/or Acknowledgments, reset the page counter to 1 and add a style parameter that specifies a Pagenumber template of "@1". (For a list of the available templates, see the section "Numbering Templates", page 407. )

### 43.5.3. Creating Title Pages

Create your title page by creating a record of type fragment and link it at the beginning of the top-level record for your book using a Contents link.

Unless you want your title page to have the same page headings as the rest of your document, you probably want to give it a Pageheading command. See the section "Specifying Page Headings and Footings", page 385.

You probably do not want it numbered, so you should adjust the page counter appropriately after the link to your title page. See the section "Handling Tables of Contents and Frontmatter", page 387.

You use formatting environments to place the text where you want it on your title page (see the section "Positioning Text in Concordia", page 129). If you have a boiler plate or copyright page, you should be sure to place a Newpage command in the appropriate place.

If your title pages are very elaborate, you might want to define a title page environment in your book design to handle them. See the Article document type in SYS:CONCORDIA;BD-ARTICLE.LISP for an example of a title page environment.

## 43.6. Modifying a Single Instance of an Environment

Sometimes you might find that you want to modify the appearance of a single instance of an environment. Please note that this is not recommended. Here's an example of why: Let's say a record is included in a dictionary and also in some reference cards. If you modify an environment in the record to format just the way you want in the dictionary, then it might look entirely wrong in the reference cards. This is an issue of modularity. See the section "Workstyle Issues in Writing with Symbolics Concordia", page 211.

If you must modify a single instance of an environment you do this by modifying the environment markup. For example, you might want to close up the space around some instance of an environment, in which case you modify the Above and Below attributes of the environment markup. Or, you might decide that the items in a particular list should be bulleted rather than numbered. In this case you replace one environment with another by modifying the name attribute of the environment markup.

Follow this procedure to edit the attributes of a single instance of an environment.

1. Use one of the following to begin the procedure:
  - Position the mouse cursor on the markup (either delimiter) and click `c-m-Right`.
  - Put the editor cursor within an environment and click on Change Environment under **Markup** in the menu.
  - Put the editor cursor within an environment and use the editor command "Change Environment".

A menu pops up showing the most common and useful attributes for that environment and their current values. (If the environment has never been modified, these are the attributes' default values.)

2. For most attributes, you simply click on the value and type in a new one. Where choices are given, as in the Blanklines attribute, the current value is shown in boldface. See the section "List of Symbolics Concordia Attributes", page 170.
3. Click on another choice to select it. For a description of the common attributes and their valid values, see the section "The Most Common Environment Attributes", page 137. The menu item Other Attributes lets you select any of the other attributes provided by Symbolics Concordia. Click on "an attribute name", then press HELP to see a complete list.

Modifications made to a single instance of an environment are local to that instance environment. In other words, changing the attributes of one Itemize envi-

```

Attributes for this environment:
Environment Name: Itemize
Above: 1 Lines
Below: 1 Lines
Blanklines: Break Hinge Hingebreak Hingekeep Ignore Ignored Kept
Indent: -2 Characters
Leftmargin: +2 Characters
Spacing: 1 Lines
Spread: 1 Lines
Other Attribute: an attribute name
Click on this line to reset attributes to default values.
<ABORT> aborts, <END> uses these values

```

Figure 61. The menu shows the current attributes of an Itemize environment.

ronment does not alter any other Itemize environment. If you need to make the same modification for more than one instance of an environment, mark the environment as a region, kill it, and yank it back at another location. (See **Kill History** in the menu.)

To modify all instances of an environment in a single book, see the section "Modifying a Single Book", page 384. To modify all instances of an environment for all books of a particular document device type, see the section "Modifying a Document Device Type", page 373.

Modified environments are denoted in the record by an ellipsis. See Figure 62.

```

Itemize ...
Itemize ...

```

Figure 62. Modified Environment Markup showing telltale ellipsis.

To return to the default values of all attributes, click on the next-to-last line of the menu. The ellipsis on the environment markup delimiters will disappear. Values of individual attributes can only be reset manually.

When you change from one environment to another (by changing the Name attribute), the default definition for the new environment is inserted in the menu.

"Query Change Environments" enables you to change instances of one environment to instances of some other environment within the current buffer. This is convenient if you suddenly realize that all your bulleted lists should be numbered lists.

## 43.7. Creating a New Document Type

In Symbolics Concordia, a document type has at least three parts: a **sage:define-book-design** form, an **sage::initialize-new-book-design** form, and a **sage:note-book-design-specifics** form.



1. The **sage:define-book-design** form defines the higher-level characteristics of the document, including its trim size and margins.
2. The **sage::initialize-new-book-design** form defines the specific attributes of the document, for example its page headings, and whether or not it is to be right justified.
3. The **sage:note-book-design-specifics** form includes the newly defined document type in the *Book Design Registry*, the list **sage::\*book-design-registry\***

Additionally a document type can include specially defined counters (for chapter numbering, for example) or heading styles, each of which can be defined as its own book design module. (For example, the Article document type includes an **article-headings** book design; see the section "Defining Headings for an Article", page 408. )

To create a new document type, you first define the document type itself, using **sage::define-document-type**. Next you use **sage:define-book-design** to link that document type with a device type. Then you write an **initialize-your-book-design** form to set up the specifics for the book design. Finally, you use **sage:note-book-design-specifics** to inform Symbolics Concordia that the new document type exists.

**sage:define-book-design** *name options &body forms* *Function*

A **sage:define-book-design** form can contain six kinds of forms: **use**, **define**, **modify**, **first**, **counters**, and **collectors**.

<b>use</b>	Takes a book design and inherits the attributes of that design. Generally you want to <b>use</b> Generic-Generic and one or more other predefined book designs. If you want to supersede some of the predefined internal book designs (see the section "Book Design Elements", page 366), you might find it easier to specify explicitly which internal book designs you want, omitting Generic-Generic. The Lisp command <b>sage:graph-book-design-users</b> is useful to check which book designs use which other book designs so that you can be sure you are omitting those you do not want included. See the function <b>sage:graph-book-design-users</b> , page 426. <b>sage:graph-book-design-uses</b> can also be used to see exactly which book designs are being included in any given book design.
<b>define</b>	Redefines some attribute(s) of the parent book designs completely or defines some environments which are exclusive to <i>name</i> . You specify all the parameters for the attribute(s).
<b>modify</b>	Modifies some attribute(s) of the parent book designs. You specify only those parameters you want to change; all others are inherited from whatever the parent design is.

- first** Sets up the top-level environment (**envr**), which defines the paper size and margins, and the environment (**init**) which calls the initialize function for the book design. The initialize form can set up page headings and footings.
- collectors** Lets you define behavior associated with the frontmatter and endmatter of a book.

Its syntax is

```
(define-book-design ()
  ...
  (collectors
   collector-spec-1
   collector-spec-2
   ...))
```

Each *collector-spec-n* looks like

*(collector-name &key name init)*

*collector-name* is a name from the set {Contents, Tables, Figures, Index}

*name* should either be a string or a symbol. If a string, it is presumed to be the "name" for that document section, and will be emitted as Prefatory text for that section. For example, the default name for Contents is "Table of Contents". If a symbol, it should name a function of no arguments that is called within the context of **sage:making-sage-directives** to produce the name for the section (see **sage:making-sage-directives**). *name* is not required. If not specified, no name appears.

*init*, if present, should be a symbol naming a function of no arguments that is called within **sage::making-sage-directives** to initialize formatter variables, etc. before this part of the document is formatted. It can change the pageheadings, change the rendering of the page counter, etc. Often it will do **NewPage UntilEven**. The default *init* for Contents is **sage::init-standard-table-of-contents**. The default *init* for Index is **sage::init-standard-index**.

Please note: The **collectors** clause lets you define behavior, not modify it. Therefore, if you want the table of contents to come out as "ToC", you need to include:

```
(contents :name "ToC"
          :init init-standard-table-of-contents)
```

**counters** Lets you define behavior associated with section headings like chapter, subsection, and the like. For example, in the book design element **sage::numbered-sections**, a counter for Chapter is defined as follows:

```
(chapter
  :numbered "@1."
  :referenced "@1"
  :title ((title counter-contents)
    (idirs
      (envr (HD1 above 2 below 2 facecode b)
        counter-contents (spaces 2) title)))
  :contents ((title counter-contents)
    (render-regular-toc
      '(Tc1) title counter-contents)))
```

In this example, the keywords specify the following:

<b>:numbered</b>	The numbering template used in the chapter opening. For examples of numbering templates, see the section "Defining Counters for an Article", page 405.
<b>:referenced</b>	The numbering template used in references to the chapter, for instance in the table of contents.
<b>:title</b>	The formatting specifications for the chapter opening. This is where the environment Hd1 is used.
<b>:contents</b>	The formatting specifications for the table of contents for chapter entries. This is where the environment Tc1 is used.

### 43.7.1. Creating a Letter Document Type

First we define the document type:

```
(define-document-type 'letter)
```

A letter has certain characteristics that are the same no matter what device you use to display it.

Now we can define those things that make a letter different from other document types: an address, a subject, a salutation, a closing, an author (the person sending it), and a few other bookkeeping items like the cc: list and record of any enclosures.

The form **sage::define-sage-environment** takes an environment name followed by a document type and a device type. The document type is Letter, and since the environment is to be for all types of letters, the device is the Generic device. The body of each form is the formatting attributes that the environment is to use. Here are two Letter environments:

```
(define-sage-environment SubjectEnv letter generic
  '((USE text)
    (below 1)
    (leftmargin "+1.75picas")
    (indent "-1.75picas")
    (afterentry "RE: ")))
(define-sage-environment SalutationEnv letter generic
  '((USE Text)
    (below 1)))
```

The Subject environment is to be a filled environment, followed by a blank line. That is specified here so that when a subject is present, it automatically spaces itself with one blank line above the salutation. If the subject is omitted, there are no extra blank lines. The subject itself is preceded by the string "RE: " and if it is more than one line long, it wraps so that the second line comes out under the beginning of the first text, not at the global left margin. The Salutation environment just prints a text string followed by a blank line.

For each of these environments, we define a command to gather the information to go in the environment. The form for the Subject command looks like this:

```
(define-sage-command subject ((subject :text))
  :text? T :ends-with-newline? T :starts-with-printing-char? T)
```

Next we tell the formatter how to handle the information from the subject command by defining a method for Subject:

```
(define-command-gbox-method
  (command-gbox-insert-generated-text Subject) ()
  (splice-contents-list self environment
    (making-sage-directives ()
      (in-environment (() SubjectEnv)
        (when subject)
          (princ subject))))
  (splice-box-out self))
```

This method associates the command Subject with the Subject environment. Not all letters have subject lines, so the subject command can be omitted from a letter. But all letters have a Salutation. The Salutation command and its associated method look like this:

```
(define-sage-command salutation (&optional (recipient :text))
  :text? T :ends-with-newline? T :starts-with-printing-char? T)
```

```
(define-command-gbox-method
  (command-gbox-insert-generated-text Salutation) ()
  (splice-contents-list self environment
    (making-sage-directives ()
      (in-environment (() SalutationEnv)
        (if (null recipient)
            (princ "Greetings:")
            (princ (string-append "Dear " recipient ":"))))))
  (splice-box-out self))
```

This says that the `Salutation` command takes an optional argument *recipient* which is a text string. The method says that if there is a recipient, that text string is printed, but if no argument is given to the `Salutation` command, then the string "Greetings:" is printed.

The other environments, commands and methods are defined similarly:

```
(define-sage-environment LetterDateEnv letter generic
  '((USE Text)
    (below 1)))
(define-sage-environment AddressEnv letter generic
  '((USE Text)
    NoFill
    (below 1)))
(define-sage-environment ClosingEnv letter generic
  '((USE Text)
    (above 1)
    (below 4)))
(define-sage-environment SignatureEnv letter generic
  '((USE Text)
    NoFill
    (below 1)))
(define-sage-environment InitialsEnv letter generic
  '((USE Text)
    NoFill))
(define-sage-environment CCEnv letter generic
  '((USE Text)
    ;NoFill
    (leftmargin "+1.75picas")
    (indent "-1.75picas")
    (afterentry "cc:  "))
(define-sage-environment EnclEnv letter generic
  '((USE Text)
    ;NoFill
    (leftmargin "+2picas")
    (indent "-2picas"))
```

```

                (afterentry "encl: "))
(define-sage-command letterdate (&optional (letterdate :text))
  :text? T :ends-with-newline? T :starts-with-printing-char? T)

(define-command-gbox-method
  (command-gbox-insert-generated-text LetterDate) ()
  (splice-contents-list self environment
    (making-sage-directives ()
      (in-environment (() LetterDateEnv)
        (if (null letterdate)
            (format-date-in-style
              (time:get-universal-time) "March 8, 1952")
              (princ letterdate))))))
  (splice-box-out self))

(define-sage-command address ((address :text))
  :text? T :ends-with-newline? nil :starts-with-printing-char? T)

(define-command-gbox-method
  (command-gbox-insert-generated-text Address) ()
  (splice-contents-list self environment
    (making-sage-directives ()
      (in-environment (() AddressEnv)
        (when address
          (princ address))))))
  (splice-box-out self))

(define-sage-command subject ((subject :text))
  :text? T :ends-with-newline? T :starts-with-printing-char? T)

(define-sage-command closing (&optional (closing :text))
  :text? T :ends-with-newline? T :starts-with-printing-char? T)

(define-command-gbox-method
  (command-gbox-insert-generated-text Closing) ()
  (splice-contents-list self environment
    (making-sage-directives ()
      (in-environment (() ClosingEnv)
        (if (null closing)
            (princ "Sincerely,")
            (princ closing))))))
  (splice-box-out self))

```

```

(define-sage-command Signature (&optional (Signature :text)
                                          (jobtitle :text))
  :text? T :ends-with-newline? T :starts-with-printing-char? T)

(define-command-gbox-method
  (command-gbox-insert-generated-text Signature) ()
  (splice-contents-list self environment
    (making-sage-directives ()
      (in-environment (() SignatureEnv)
        (when Signature
          (princ Signature))
        (when jobtitle
          (terpri)
          (princ jobtitle))))))
  (splice-box-out self))

(define-sage-command Notations (&key (initials :text)
                                       (cc :text)
                                       (encl :text))
  :text? T :ends-with-newline? t :starts-with-printing-char? T)

(define-command-gbox-method
  (command-gbox-insert-generated-text Notations) ()
  (splice-contents-list self environment
    (making-sage-directives ()
      (when initials
        (in-environment (() InitialsEnv)
          (princ initials)))
      (when cc
        (in-environment (() CCEnv)
          (princ cc)))
      (when encl
        (in-environment (() EnclEnv)
          (princ encl))))))
  (splice-box-out self))

```

**Note:** When you name a command or environment, do not use a hyphen in the name. Hyphenated names are for internal commands or environments, and you cannot insert such an environment or command in a .sab file using Create Markup. You must use `c-U s-M` to insert them.

Now we define a generic letter, a linking of the document type Letter with the device Generic:

```
(define-book-design letter-generic ()
  (use generic-generic)
  (modify (Hd1
           (pagebreak off)))
  (first
   (init initialize-letter-generic)))
(note-book-design-specifics 'letter :generic 'letter-generic)
```

The form **sage:note-book-design-specifics** associates the previously defined Letter document type with the universal parent device generic. Note that letter-generic *uses* generic-generic. The generic-generic book design contains all the standard formatting environments. This ensures that the letter book design has access to all of these standard environments, as well as the special ones we have defined for it. We do not want "chapters" to be treated the same in a letter as they are in a generic document, that is we do not want them to start on a new page, so we modify Hd1, the environment used to print the chapter title, to not do a pagebreak.

We also define an initialization form for the generic letter:

```
(defun initialize-letter-generic ()
  (macrolet ((in (&body body)
              '(making-sage-directives () (idirs ,@body))))
    (sage-command () 'style '((date "March 1952")))
    (sage-command () 'style '((hyphenation T) (hyphenbreak T)))
    (sage-command () 'pageheading
                     '((immediate nil)
                       (left ())
                       (right ,(in (value page)))
                       (center ())))))
```

This does not have to do very much. It determines the style for printing the date, and specifies that the page number is printed in the right hand corner of the page, on the second and subsequent pages.

We can now define how the Letter looks when displayed on the screen and when formatted and printed out.

```
(define-book-design letter-screen ()
  (use generic-screen)
  (use letter-generic))
(note-book-design-specifics 'letter :screen 'letter-screen)
```

Letter-screen uses the book designs generic-screen and letter-generic, ensuring that it has access to all the usual things that device screen does, and all the specific letter environments. The universal parent book design, generic-generic is included because it is used by letter-generic. (In fact, it is probably used by generic-screen as well.)

Notice that we do not need to have an **initialize-letter-screen** form because the initialization for the generic letter suffices.

The printed letter is a bit more complicated, however, since the limitations of the printing device and paper size must be considered. The only hardcopy device cur-



rently supported by Symbolics Concordia is the LGP2, so we define letter-lgp2:

```
(define-book-design letter-lgp2 ()
  (use generic-lgp2)
  (use letter-generic)
  (modify
    (box (BoxFlushRight Yes))
    (bodystyle (spacing 11pts) (spread 2pts)))
  (first
    ;;Style
    (envr (text
      (Paper-Width "8.5in")
      (LeftMargin "8picas")
      (RightMargin "8picas")
      (TopMargin "8picas")
      (BottomMargin "6picas"))))
    ;;Top-level begin
    (envr (Text
      (Indent "0")
      (Use BodyStyle)
      (Spaces Compact)
      (Font BodyFont)
      (FaceCode R)
      (Justification off)
      (Hyphenation On)
      (Spaces Compact))))
  ))
(note-book-design-specifics 'letter :lgp2 'letter-lgp2)
```

Letter-lgp2 uses generic-lgp2 and letter-generic but it must then set up the margins and other style parameters for the letter's appearance. This is done inside the **first** form, using two **text** environments. The first sets up the paper-width and the global margins. There are four parameters for margins, **LeftMargin**, **RightMargin**, **TopMargin**, **BottomMargin**, and **LineWidth**. Only four of them can be specified at one time. Here we specify the margins, omitting the linewidth. The second **text** environment sets up page headings and other global characteristics of the document, such as modifications to standard environments, and special environment definitions.

### 43.7.2. Creating Your Own Crossreference Appearance

You can define your own crossreference with the macro **sage:define-crossreference-appearance**, which allows you to define how a crossreference should display, including any automatically pre-pended or appended sentence fragments. Here is a definition for a crossreference appearance:

```
(sage::define-crossreference-appearance
  Experimental "An experimental appearance"
  ((quote-p t :documentation "Surround with quotes"
    :pretty-name "Quote?"))

  :formatter ((record-group contents-list)
    (ignore record-group)
    (when quote-p
      (write-char #\")))
    (sage-contents-list t contents-list)
    (when quote-p
      (write-char #\")))
  :editor ((record-group stream)
    (ignore quote-p)
    (format stream "Experimental view of ~A" record-group)))
```

This defines a new crossreference appearance called "Experimental". If you create a crossreference link and give it appearance "Experimental", you also have the option of specifying Quote? or not (in the menu). If Quote? is on, the link looks like

"The Input Editor Program Interface"

on paper. If quote-p is off, the link looks like

The Input Editor Program Interface

on paper. In the editor, it always looks like

◀ Experimental view of "The Input Editor Program Interface" Section

**sage:define-crossreference-appearance** *appearance-name* *documentation* *booleans* &key *:formatter* *:editor* *Macro*

Defines the appearance of a crossreference link type.

*appearance-name* The name for this appearance. It should be a symbol. This name (string-capitalized) will appear in appearance name menu you get when editing a link.

*documentation* A string providing brief documentation for this link. It will appear when the mouse is moved over this link type in the menu.

*booleans* A set of options to this link type. Each option is of the form *option* or (*option default* &key *:documentation* *:pretty-name*)

*option* is a symbol naming the option.

*default* is the default value. Since these are booleans, the default must be **t** or **nil**.

*:documentation* is a string providing brief documentation of the purpose of this option. It will appear when the mouse is moved over this option in the menu.

*:pretty-name* is a string which appears in place of this option name in the menu.

### **:formatter**

A form whose **car** is an argument list and **cdr** is a functional body. The body can reference the variables in the argument list, as well as any of the booleans. The two arguments passed to this function are a record-group and a "contents-list".

So it is *((record-group contents-list) &body body)*

The *record-group* is the particular record-group this link points to. The *contents-list* is a sage data structure which corresponds to the formatted name of that *record-group*.

Although **:formatter** is a keyword argument, it is required.

The formatter should do output to **\*standard-output\***. It is called within the scope of a **sage:sage-formatting**, so it can use things like **sage:sage-paragraph** or **sage:in-environment**. If it wants to output the formatted name of the target of the link, it can use **(sage:sage-contents-list t contents-list)**.

The output from formatter is what appears in place of the link when the sage formatter runs. So the formatter for the See appearance outputs things like "See the ..."

### **:editor**

A form whose **car** is an arglist and **cdr** is a functional body. The body can reference the variables in the arglist, as well as any of the booleans. The two arguments passed to this function are a record-group and a stream.

So it is *((record-group stream) &body body)*

The editor should do output to stream. This is how the link displays itself in the editor.

If there is no editor, the formatter is run and its output is what shows up in the editor.

### **43.7.3. Formatting Tables of Contents and Lists of Figures**

You control whether or not something goes into the Table of Contents in the Counter definitions. The actual appearance of the components of tables of contents (that is what font is used the counter and the title, the indentation for a given level and the interline spacing) is controlled by the book design **table-of-contents-environments**.

```

(define-book-design table-of-contents-environments ()
  (define
    (ContentsStyle (Spacing 1) (Spread "0.5lines") (Font TitleFont6))
    (TCX (Use ContentsStyle)
      ;; see environment-specially-hacks-paragraphs
      ;; which tries to figure out whether pseudo-paragraph-breaks
      ;; should be spliced in.
      (LeftMargin "+.0001cm")
      (RightMargin 5)
      (Indent 0)
      (Spread "0lines")
      (Hyphenation Off)
      (Above 0)
      (Below 0)
      Break
      Fill
      (Justification Off)
      (Spaces Compact)
      (Need "2lines"))
    (Tc0 (Use TcX)
      ;Centered
      (Above "1line")
      (Below "0.5line")
      (Need "2lines")
      (TabExport False))
    (Tc1 (Use TcX)
      (TabExport False)
      (Above "1line")
      (Below "0.5line")
      (Need "2lines"))
    (Tc2 (Use TcX) (LeftMargin "+4")
      (TabExport False)
      (Font TitleFont7)
      (Need "2lines"))
    (Tc3 (Use TcX) (LeftMargin "+9") (Font TitleFont7) (Need "2lines"))
    (TcC (Use TcX) (Font TitleFont7) (TabExport False) (Need "2lines"))
    (PermutedIndexEnv (use Format)
      (use NoteStyle)
      (longlines Wrap)
      (indent "20picas")
      (spacing ".8 lines"))
    (PermutedIndexHeaderEnv (use MajorHeading)
      Flushleft
      (above "1")
      (below "0"))
  ))

```

The final assembly of the table of contents from its components is done by **sage::render-regular-toc**. This is an internal function that is called by the **:contents** keyword in the counter definition. Here is the definition for **sage::render-regular-toc**:

```
(defun render-regular-toc (environment title counter-contents)
  (in-environment1 (() environment)
    (in-presentation (()
      :type 'page-number
      :object (filtered-string-from-contents-list
        (counter-value-contents-list (lookup-ambient-value 'page)
          :reference))
      :presentation-options '(:single-box t))
    (idirs
      counter-contents
      (spaces 2)
      (command dynamic-left-margin)
      (command rfstr
        (ncons (making-sage-directives ()
          (sage-command () 'counter-value-as-number
            (list
              (clone-counter-and-lineage
                (lookup-ambient-value 'page)))))))
      title))))
```

The actual formatting is done in the **sage:idirs** clause of this form. This one says that the counter contents (that is the chapter or section number) appears followed by two spaces. The margin is reset at that new location (for handling long titles that wrap). The page number is flushright (**rfstr**, or Right Flush STRing), and the title is formatted between the reset margin to the page number, wrapping if necessary.

If you want to have your tables of contents look different from the way Symbolics's tables of contents look, you can modify the **tcn** environments in your book design. If you want to completely override the appearance of the Symbolics style tables of contents, you can supply your own **table-of-contents-environments** book design. If you need to change the final formatting, you can write your own **render-your-own-toc** form and call it in your counter definitions.

#### 43.7.4. Formatting Page Headings

If you are unable to get the effect you want with simply using the **pageheading** and **pagefooting** sage commands in your **initialize-your-book-design** form, you can modify the various **hdn** environments in your book design. The attributes of these heading environments are defined in the **page-heading-environments** book designs. If you want to completely override the Symbolics style headings, you can supply your own version of this book design.

## 43.8. Creating Your Own 3Symanual Document Type

Suppose you wanted to create your own version of **3symanual-lgp2** that had one-inch margins all around and slightly different pageheadings. This is an absurdly simple example, since you could most easily accomplish that by resetting the margins in a transparent environment around the contents of the top-level record and adding a pageheading command. However, it serves as an example of how you can create a simple book design by making a copy of an existing one and modifying it:

```
(define-book-design My-3symanual-lgp2 ()
  (use generic-lgp2)
  (use 3symanual-lgp2)
  (first
    (envr (text
      (Paper-Width "8.5in")
      (LeftMargin "10picas")
      (RightMargin "10picas")
      (TopMargin "10picas")
      (BottomMargin "10picas"))))
    (envr (text (Indent "0")
      (Use BodyStyle)
      (justification off)
      (Spaces Compact)
      (Font BodyFont)
      (FaceCode R)))
    (init
      initialize-my-3symanual-lgp2))
  )

(note-book-design-specifics 'my-3symanual :lgp2 'my-3symanual-lgp2)

(defun initialize-my-3symanual-lgp2 ()
  (macrolet ((in-boxes (&body body)
    '(in-boxes-1 (named-lambda in-boxes () (idirs ,@body))))
    (in (&body body)
      '(making-sage-directives () (idirs ,@body))))
    (flet ((in-boxes-1 (continuation)
      (declare (sys:downward-funarg continuation))
      (making-sage-directives ()
        (in-environment (T transparent boxtype StandardInvisibleOverline)
          (in-environment (T transparent boxtype StandardOverline)
```

```

        (funcall continuation))))))
(sage-command () 'pageheading
  '((odd T) (immediate T)
    (left ,(in (value confidential)))
    (right ,(in (value page)))
    (center ())
    (line ,(in-boxes
      (value doc-title)
      (command collect-right-flushing)
      (value chapter)
      (command literal-space))))))
(sage-command () 'pageheading
  '((Even T) (immediate T)
    (Left ,(in (value page)))
    (right ,(in (value confidential)))
    (center ())
    (line ,(in-boxes (value chapter)
      (command collect-right-flushing)
      (command literal-space))))))
(sage-command () 'pagefooting '((immediate T)))
(sage-command () 'style '((date "March 1952")))
(sage-command () 'style '((hyphenation T) (hyphenbreak T)))
)))

```

## 43.9. Creating an Article Document Type

First you define an article document type.

```
(define-document-type 'article)
```

Then you create your own book design using the form **sage:define-book-design**. An article is usually a document primarily for hardcopy distribution, so you probably want to create a document design using the LGP2.

```
(define-book-design article-lgp2 () ...)
```

An article has many of the same attributes as a manual, so it can be patterned on the existing 3symanual document design.

```
(define-book-design article-lgp2 ()
  (use 3symanual-lgp2) ...)
```

3symanual-lgp2 uses Generic-Generic and Generic-Lgp2, so you have all the generic environments available. The handling of sections in articles is usually a little different from a manual, however, so you can define the special counters to use.

### 43.9.1. Defining Counters for an Article

Each counter has a template associated with it that determines how it is numbered, how it is referenced, how its title should appear in the text, and what format it should follow in the table of contents. Allowing any one of these things to

be **nil**, that is the empty string, "", means that the counter does not have that attribute.

Here is the chapter counter as defined for 3symanual:

```
(chapter
  :numbered "@1."
  :referenced "@1"
  :title ((title counter-contents)
    (idirs
      (envr (HD1 above 2 below 2 facecode b)
        counter-contents (spaces 2) title)))
  :contents ((title counter-contents)
    (render-regular-toc '(Tc1) title counter-contents)))
```

And here is the chapter counter for an article:

```
(chapter
  :numbered "@1."
  :referenced "@1"
  :title ((title counter-contents)
    (idirs (envr HD1 counter-contents (spaces 2) title)))
  :contents ((title counter-contents)
    ;;---
    (render-regular-toc '(Tc1) title counter-contents)))
```

Notice that the chapter opening for 3symanual prints its title in text, as defined by the **:title** keyword, set off from surrounding text by 2 lines above and below, and uses the header environment **HD1**, boldface. The article, on the other hand, has a title that is only set off from surrounding text by the standard spread, and uses the default facecode for **HD1**.

The attribute **:within** before the **:numbered** attribute means that the numbers are to be relative to some other counter, for example numbering sections **:within chapter** means that the sections are numbered 1.1, 1.2, 1.3, ..., 1.*n*. Sections are numbered within chapters, and subsections are numbered within sections.



```

(define-book-design article-counters ()
  (counters
    (chapter
      :numbered "@1."
      :referenced "@1"
      :title ((title counter-contents)
              (idirs (envr HD1 counter-contents (spaces 2) title)))
      :contents ((title counter-contents)
                 ;;---
                 (render-regular-toc '(Tc1) title counter-contents))
      )
    (section
      :within chapter
      :numbered "@#.01."
      :referenced "@#.01"
      :title ((title counter-contents)
              (idirs (envr HD2 counter-contents (spaces 2) title)))
      :contents ((title counter-contents)
                 ;;---
                 (render-regular-toc '(Tc2) title counter-contents))
      )
    (subsection
      :within section
      :numbered "@#.01."
      :referenced "@#.01"
      :title ((title counter-contents)
              (idirs (envr HD3 counter-contents (spaces 2) title)))
      :contents ((title counter-contents)
                 ;;---
                 (render-regular-toc '(Tc3) title counter-contents))
      )
    )
  ))

```

#### 43.9.1.1. Numbering Templates

The available numbering templates are:

<i>Description</i>	<i>Template</i>	<i>Prints As</i>
Arabic numerals	1	1,2,3,...
Lower case roman	i	i,ii,iii,...
Upper case roman	I	I,II,III,...
Lower case alphabetic	a	a,b,c,...
Upper case alphabetic	A	A,B,C,...
Ordinal numbers	F	First, Second, Third,...
Lower case ordinals	f	first, second, third,...
Cardinal numbers	O	One, Two, Three,...
Lower case cardinals	o	one, two, three,...

To print numbers within a parent counter, use `@#` to represent the parent counter. For example, if chapters are numbered using arabic numerals, and you specify that sections are to be numbers with lower case alphabetic, `@#.a` specifies that sections are to print as 1.a, 1.b, and so on.

Now, you use this book design, **article-counters**, in the article book design:

```
(define-book-design article-1gp2 ()
  (use 3smanual-1gp2)
  (use article-counters)...)

```

### 43.9.2. Defining Headings for an Article

Heading environments are modified to not cause pagebreaks in an article.

```
(define-book-design article-headings ()
  (define
    (FTG (FONT smallbodyfont)
      (FACECODE i)
      (FIXED "-0.5inch")
      NOFILL
      UNNUMBERED
      (UNDERLINE off)
      (SPACING 1)
      (USE I)
      (TABEXPORT False)
      (COLUMNS 1)
      (COLUMNMARGIN 0)
      (CAPITALIZED off)
      (SPREAD 0)
      (INDENT 0)
    )
  )

```

```

        (AfterEntry "@tabclear()"))
(modify
;;Section heading environments.
(HDX (Use BodyStyle)
      (font titlefont2)
      (Spacing 11pts)
      (Need 10)
      (PageBreak Off)
      (Above 2.0)
      (Below 1.5)
      (LeftMargin 0))
(Hd1 (Use HdX)
      (FaceCode B)
      (Capitalized On))
(Hd2 (Use HdX)
      (FaceCode B)
      (Capitalized Off))
(Hd3 (Use HdX)
      (FaceCode B)
      (Capitalized Off))
(ContentsStyle (Spacing 1)
                (Spread 1)
                (Font BodyFont))
;;Table of Contents environments
(TCX (Use ContentsStyle)
      (FaceCode R)
      (RightMargin 4)
      (indent 0)
      (hyphenation off)
      (above 0)
      (below 0)
      break
      fill
      (justification off)
      (spaces compact)
      (need 2lines)
      )
(Tc1 (Use TcX)
      (FaceCode B) (Above 0.5))
(Tc2 (Use TcX)
      (LeftMargin "+2picas"))
(Tc3 (Use TcX)
      (LeftMargin "+4picas"))
))

```

Now we have:

```
(define-book-design article-lgp2 ()
  (use 3symanual-lgp2)
  (use article-counters)...)

```

Here is the complete **article-lgp2** book design. You can find it, and some special environments defined for articles, in `SYS:CONCORDIA:BD-ARTICLE.LISP`.

```
(define-book-design article-lgp2 ()
  (use 3symanual-lgp2)
  (use article-counters)
  (use article-headings)
  (first
    ;;Style
    (envr (text
      (Paper-Width "8.5in")
      (LeftMargin "1.33inches")
      (RightMargin "1.33inches")
      (TopMargin "1.33inches")
      (BottomMargin "1inch"))))
    (envr (text (Indent 4)
      (Use BodyStyle)
      (spacing 2)
      (spread 0)
      (Spaces Compact)
      (Font BodyFont)
      (FaceCode R)))
    (init
      initialize-article-lgp2)
  ))

(note-book-design-specifics 'article :lgp2 'article-lgp2)

(defun initialize-article-lgp2 ()
  (sage-command () 'style '((date "March 1952")))
  (sage-command () 'style '((hyphenation nil) (hyphenbreak nil)))
  )

```

## 43.10. Defining a Book Design System for a Site

When you have created a book design, you can use it by evaluating its definition, mentioning it as the **:document-type** keyword in the **sage::register-book** form for your document, and evaluating that. Then using Format Pages on your document in the Page Previewer uses your design. However, if you want your book design to be accessible to everyone at your site or to distribute it with your documentation system, you should install it in your system.

To install a book design in your system, make it a component of your documentation system. You have two choices:

1. You can make your extensions into a system and include that system as a module of **:type system** in your documentation system definition.
2. You can include the file they are defined in as a module of **:type lisp** in your documentation system definition.

Presumably, a large library of special-purpose formatting environments, record types, and book designs would be more easily handled as a system, while some minor modifications would be handled by just having a Lisp module in the documentation system. In any case, the non-Sage modules should be placed before the Sage modules, and their loading dependencies should be **:serial**. For more information about system declarations for documentation systems, see the section "The System Declaration for a Documentation System", page 431.

Here is an example of a system definition for a documentation system that defines some special record types:

```
(sage:define-documentation-system sched-doc
  (:pretty-name "Scheduling and Planning Documentation"
   :default-pathname "sys:sched;doc;"
   :patchable t
   :advertised-in (:herald)
   :maintaining-sites (:down-under)
   :distribute-sources t
   :distribute-binaries nil
   :default-module-type :sage
  )

  (:module record-types "record-types" (:type :lisp))
  (:module books "book-registry.lisp" (:type :lisp))
  (:module schedule ("scheduler" "calendar" "notify"))
  (:module plan ("long" "short"))
  (:module installation ("install"))
  (:module illustrations ("sched-pictures.pic")
   (:type :data-no-load)
   (:source-category :restricted))
  (:serial record-types books
   (:parallel schedule plan installation ...))
)
```

## 43.11. Dictionary of Book Design Browser Commands

### 43.11.1. Clear Display

Clears the display pane of the Book Design Browser. The display is not actually lost. You can use the scroll bar to the left of the Display Pane to scroll back to previous output. Output for each command is separated by a line of dashes.

### 43.11.2. Describe Book Design Element

Describe Book Design Element *book-design*

Lists other book design elements from which *book-design* inherits. In other words, this command lists *book-design*'s parent and grandparent book design elements. See the section "Inheritance Dependency in Symbolics Concordia Book Design", page 359. The command also includes a brief description of what the element does.

```
USER-VISIBLE-ENVIRONMENTS is a book design element.
```

```
  Defines or modifies some environments.  
  Inherits from:  
    INNER-ENVIRONMENTS
```

Figure 63. Describe Book Design Element user-visible-environments

If *book-design* is a top-level book design element, then the command lists the top-level style and top-level text clauses, which includes top-level specifications, and the init clause function name.

**3SYMANUAL-LGP2** is a top-level book design element.

Defines or modifies some environments.

Inherits from:

GENERIC-LGP2

**GENERIC-PAPER**

GENERIC-GENERIC

PAGE-HEADING-ENVIRONMENTS

SECTION-HEADING-ENVIRONMENTS

TABLE-OF-CONTENTS-ENVIRONMENTS

NUMBERED-SECTIONS

OTHER-COUNTERS

STANDARD-COLLECTORS

USER-VISIBLE-ENVIRONMENTS

INNER-ENVIRONMENTS

FACECODE-ENVIRONMENTS

DEFLINE

LGP2-INNER-STYLES

DEFLINE

Top-level Style:

PAPER-WIDTH 8.5in

LEFTMARGIN 8picas

RIGHTMARGIN 8picas

TOPMARGIN 8picas

BOTTOMMARGIN 6picas

Top-level Text:

INDENT 0

USE BODYSTYLE

SPACES COMPACT

FONT BODYFONT

FACECODE R

Init: INITIALIZE-3SYMANUAL-LGP2

Figure 64. Describe Book Design Element 3symanual-lgp2

Use this command to investigate the top-level specifications for a document device type (once you've found the top-level book design element for the given document device type). See the section "Describe Document Device Type", page 413. See the section "An Example of a Modification to a Top-level Specification in a Document Device Type", page 380.

You can also use this command to investigate the book design elements which contribute to the definition of an environment. See the section "Describe Environment", page 414. See the section "Examples of Document Device Type Modifications", page 374.

### 43.11.3. Describe Document Device Type

Describe Document Device Type *document-device-type*

Lists the book design elements which make up *document-device-type*.

**3SYMANUAL LGP2** (a document device type) is specified by:

```

3SYMANUAL-LGP2
  GENERIC-LGP2
    GENERIC-PAPER
      GENERIC-GENERIC
        PAGE-HEADING-ENVIRONMENTS
        SECTION-HEADING-ENVIRONMENTS
        TABLE-OF-CONTENTS-ENVIRONMENTS
        NUMBERED-SECTIONS
        OTHER-COUNTERS
        STANDARD-COLLECTORS
        USER-VISIBLE-ENVIRONMENTS
          INNER-ENVIRONMENTS
        FACECODE-ENVIRONMENTS
        DEFLINE
      LGP2-INNER-STYLES
    DEFLINE
  
```

Figure 65. Describe Document Device Type 3symanual LGP2

The first book design element listed is the top-level element for *document-device-type*. The other elements are those which the top-level element inherits from, that is, its parents. The indentation indicates which elements inherit from which other elements. Therefore, in this example, **sage::3symanual-lgp2** (child) inherits from **sage::generic-lgp2** (parent) which in turn inherits from **sage::generic-paper** (grandparent).

Boldfaced elements set top-level specifications.

Use this command when you want to make a modification to the top-level specifications for a document device type. See the section "An Example of a Modification to a Top-level Specification in a Document Device Type", page 380.

#### 43.11.4. Describe Environment

Describe Environment *environment document-device-type*

Displays a table of the attributes and their values which make up the default definition for *environment* in *document-device-type*. See the section "Dictionary of Symbolics Concordia Markup Environments and Commands", page 141.

This table also includes environments and book design elements which are *environment's* parent environments and book design elements. See the section "Inheritance Dependency in Symbolics Concordia Book Design", page 359.



**Hd2** (an environment) is defined in **3SYMANUAL LGP2** (a document device type) as follows:

Attribute	Value	From Environment	In Book Design Element	Inherited by Book Design Element
Font	TITLEFONT3	Hd2	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Above	1cm	Hd2	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Below	0.7cm	Hd2	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Need	1.2 inches	Hd2	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
LeftMargin	+ .0001cm	HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Indent	0	HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Break		HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Fill		HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Justification	OFF	HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Spaces	COMPACT	HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
FaceCode	R	HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
GroupNext		HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
VerticalJustificationAllowed	ABOVE	HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Hyphenation	OFF	HDX	<b>SECTION-HEADING-ENVIRONMENTS</b>	GENERIC-GENERIC
Spacing	1	TitleStyle	<b>LGP2-INNER-STYLES</b>	GENERIC-LGP2
Spread	0.51lines	TitleStyle	<b>LGP2-INNER-STYLES</b>	GENERIC-LGP2

**Hd2** (an environment) is inherited by:

Environment	In Book Design Element
Heading	APPROACH-ENVR-MODS

Figure 66. Describe Environment Example 3symanual LGP2

Column #/name	Description
1/Attribute	The names of the attributes which make up the definition of <i>environment</i> in <i>document-device-type</i> .
2/Value	The value of the attribute in column 1. If this slot is empty, then the value of that attribute is "On" or "Yes". If the value is " <i>pageheading</i> " or " <i>pagefooting</i> ", then the actual value is too lengthy to be of use in the display. In that case, you can see the full attribute value in the source file.
3/From Environment	The environment which donates the attribute in column 1 to the definition of <i>environment</i> for <i>document-device-type</i> .
4/In Book Design Element	The element which contains the environment in column 3. Boldface type indicates that the environment in column 3 is defined in this element; italics indicates that the environment in column 3 is modified in this element.
5/Inherited by Book Design Element	The element which inherits the element in column 4.

In addition, after the table, the command lists any environments which are *environment's* children (that is those environments which inherit from and are therefore dependent upon *environment's* definition) and the book design elements in which they are defined.

The display shows the complete inheritance of *environment* for *document-device-type*. Many entries in this display are mouse sensitive. See the mouse documentation line for instructions.

Use this command to investigate the definition of an environment for a document device type. See the section "An Example of a Simple Modification to an Environment in a Document Device Type", page 374. See the section "An Example of a Modification to an Environment with Dependencies in a Document Device Type", page 378.

#### 43.11.5. Edit Book Design Element

Edit Book Design Element *book-design*

Selects the Zmacs editor, reads the source file for *book-design* into an editor buffer (if it is not already read in), selects the buffer, and positions the cursor at the beginning of the **sage:define-book-design** form for *book-design*.

See the section "Modifying a Document Device Type", page 373.

#### 43.11.6. List Book Design Elements Using Environment

List Book Design Elements Using Environment *environment*

Lists the book design elements which define and/or modify *environment*.

**Description** (an environment) is *defined* by the following book design element:

```
USER-VISIBLE-ENVIRONMENTS
```

**Description** (an environment) is *modified* by the following book design elements:

```
LISP-DICTIONARY-LGP2
APPROACH-ENVR-MODS
```

Figure 67. List Book Design Elements Using Environment Description

#### 43.11.7. Show Dependencies on Book Design Element

Show Dependencies On Book Design Element *book-design*

Lists book design elements which inherit *book-design*. In other words, this command lists *book-design*'s children and grandchildren book design elements. See the section "Inheritance Dependency in Symbolics Concordia Book Design", page 359.

**USER-VISIBLE-ENVIRONMENTS** (a book design element) is inherited by:

```

GENERIC-GENERIC
  GENERIC-SCREEN
    GENERIC-DEX
      GENERIC-DEX-BACKGROUND
    MEMO-SCREEN
    LETTER-SCREEN
    GENERIC-MAC-SCREEN
  GENERIC-PAPER
    GENERIC-LGP2
      3SYMANUAL-LGP2
        LISP-DICTIONARY-LGP2
        ARTICLE-LGP2
        DOC-EX-LGP2
        MAC-DOC-LGP2
      APPROACH-LGP2
      MEMO-LGP2
      LETTER-LGP2
      INSTALLATION-LGP2
    GENERIC-DMP1
  MEMO-GENERIC
    MEMO-LGP2
    MEMO-SCREEN
  LETTER-GENERIC
    LETTER-SCREEN
    LETTER-LGP2
  ARTICLE-GENERIC

```

Figure 68. Show Dependencies On Book Design Element user-visible-environments

This example shows that **sage::user-visible-environments** (parent) is inherited by **sage::generic-generic** (child) which in turn is inherited by **sage::generic-screen**, **sage::generic-paper**, and so on (grandchildren). **sage::generic-paper** (parent), in turn, is inherited by **sage::generic-lgp2** (child) which is inherited by **sage::3symanual-lgp2** (grandchild).

#### 43.11.8. Show Dependencies on Environment

Show Dependencies On Environment *environment*

Lists environments which inherit *environment*. In other words, this command lists *environment's* children. See the section "Inheritance Dependency in Symbolics Concordia Book Design", page 359.

**Itemize** (an environment) is inherited by:

```

Environment In Book Design Element _____
Enumerate  APPROACH-ENVR-MODS
Checklist  USER-VISIBLE-ENVIRONMENTS
Enumerate  USER-VISIBLE-ENVIRONMENTS
Checklist  3SYMANUAL-LGP2

```

Figure 69. Show Dependencies On Environment Itemize

## **43.12. Book Design Functions Dictionary**

### **43.12.1. Internally Used Formatting Styles**

#### **43.12.1.1. Abovebelowstyle Environment**

The basic environment parameters for the spacing around any environment. The default in the Generic-Generic document type is one line above and one below.

#### **43.12.1.2. Bodystyle Environment**

The basic environment parameters for the body of the document, that is the text. The default in the Generic-Generic document type is to use BodyFont, with spacing 13 points and spread 6 points.

#### **43.12.1.3. Contentsstyle Environment**

The basic environment parameters for formatting the frontmatter. The default in the Generic-Generic document is spacing 1 line, spread 0.5 lines, using TitleFont6.

#### **43.12.1.4. Largestyle Environment**

An environment for text in a large font. Its default definition in the Generic-Generic document design is spacing 1 line, spread 0.5 lines, using TitleFont5. It is not used in any of the predefined document designs.

#### **43.12.1.5. Notestyle Environment**

An environment for formatting notes, as at the end of a chapter, for example. Its default definition in the Generic-Generic document type is spacing 1 line, spread 0.5 lines, using SmallBodyFont, FaceCode R. It is not used in any of the predefined document types.

#### **43.12.1.6. Titlestyle Environment**

The most basic environment parameters for headings and table of contents entries. The defaults in Generic-Generic are spacing 1 line, spread 0.5 lines, using TitleFont3.

#### **43.12.1.7. Fnenv Environment**

The environment used for footnotes. Its definition in the Generic-Generic document design is:

```
(define-sage-environment FnEnv generic generic
  (:pretty-name "FnEnv" :internal t)
  '((Use Text)
    (Counter FootnoteCounter)
    (Font SmallBodyFont)
    (Above 1)
    Foot
    Crspace
    (Use R)
    (TabExport False)
    (LeftMargin 0)
    (RightMargin 0)
    (Indent 2)
    (Spread 1)
    UnNumbered
    (Spacing 1)
    (Break off)))
```

#### 43.12.1.8. Indexenv Environment

The `Indexenv` controls the appearance of the index. Here is its definition in the `Lgp2-Inner-Styles` Book Design:

```
(IndexEnv Break
  CrBreak
  Fill
  (BlankLines Kept)
  (Justification Off)
  (Columns 1)
  (ColumnMargin 0.4in)
  Boxed
  (Above 2)
  (Hyphenation Off)
  (LineWidth "2.6in")
  (Use NoteStyle)
  (Spread "0lines")
  (Spaces Kept)
  (LineWidth "2inches")
  (LeftMargin "+8")
  (Indent "-8"))
```

The style of the index is controlled by the Style Parameter, **indexstyle**.

#### IndexStyle

Controls the style of index used by Concordia. It has two values, `MultiLevel`, the standard index, and `Permuted`, the style used in *Genera Workbook* and *Genera User's Guide*. The default is `MultiLevel`. To produce books that have `Permuted` indexes, reset this style parameter in your Book Design by adding `(indexstyle permuted)` to your **Initialize-your-book-design-LGP2** form.

## **43.12.2. Internally Used Heading Styles**

### **43.12.2.1. Hdg Environment**

The environment used in pageheadings. In the Generic-Generic document design it uses `SmallBodyFont` and `FaceCode R`.

### **43.12.2.2. FTG Environment**

The environment used in pagefootings. In the Generic-Generic document design it uses `SmallBodyFont` and `FaceCode R`.

### **43.12.2.3. Hd0 Environment**

The environment used by Majorheadings. In the default Generic-Generic document design, it uses `TitleFont5`, is centered, has 3 inches of blank space above it and 2 inches below it, and always starts on an odd numbered page.

### **43.12.2.4. Hd1 Environment**

The environment used for chapter openings. In the default Generic-Generic document design, it uses `TitleFont5`, is flushleft, has 0.5 inches of blank space below it, and always starts on an odd numbered page.

### **43.12.2.5. Hd1a Environment**

The environment used for Appendix titles. It is exactly like `Hd1` except that it is centered. See the section "`Hd1 Environment`", page 420.

### **43.12.2.6. Hd2 Environment**

The environment used for section titles. In the Generic-Generic document design it uses `TitleFont3`, has 1 centimeter of blank space above it and 0.7 centimeters below it, and requires that there be at least 1.2 inches of space on the page, which insures that the first line of the text can be placed on the same page as the section heading.

### **43.12.2.7. Hd3 Environment**

The environment used for subsection titles. In the Generic-Generic document design it uses `TitleFont2`, has 0.25 inches of blank space above it and 0.16 inches below it, and requires that there be at least 0.8 inches of space on the page.

### **43.12.2.8. Hd4 Environment**

The environment used for subsubsection titles. In the Generic-Generic document design it uses `TitleFont2`, has 0.15 inches of blank space above it and 0.1 inches below it, and requires that there be at least 0.7 inches of space on the page.

See also, the user-available environments, `Majorheading`, `Heading`, and `Subheading`.

### 43.12.3. Internally Used Table of Contents Entry Styles

#### 43.12.3.1. Tcx Environment

The basic environment for Table of Contents and other frontmatter (lists of figures, for example) entries. It sets up the basic environment for all entries. It is used by all the other contents style environments.

#### 43.12.3.2. Tc0 Environment

The environment used to format Table of Contents entries of things formatted with the Hd0 environment. See the section "Hd0 Environment", page 420.

#### 43.12.3.3. Tc1 Environment

The environment used to format Table of Contents entries of things formatted with the Hd1 environment. See the section "Hd1 Environment", page 420.

#### 43.12.3.4. Tc3 Environment

The environment used to format Table of Contents entries of things formatted with the Hd3 environment. See the section "Hd3 Environment", page 420.

#### 43.12.3.5. Tcc Environment

The environment used for formatting the Table of Contents entries of the Lists of Figures and the List of Tables.

### 43.12.4. Book Design Functions and Variables

**sage::\*allow-index-commands-from-record-titles\*** *Variable*  
 Controls whether or not all record names become index entries. The default is **t**, all record names provide index entries. If you have indexed your document using Index Primary and Index Secondary commands (see the section "Creating an Index in a Symbolics Concordia Document", page 187), you might want to turn off the automatic indexing. In that case set this variable to **nil** in your book design.

**sage:\*default-device-type\*** *Variable*  
 Specifies the device type for which to format a topic. There are two devices currently supported by Symbolics Concordia, SCREEN and LGP2. SCREEN is the default for online formatting. LGP2 is the default for hardcopy.

**sage:\*default-document-type\*** *Variable*  
 Controls what book design is used for formatting when no **sage::register-book** form has been evaluated for the topic. The default is GENERIC for online formatting and 3SYMANUAL for hardcopy formatting. Your book design system can set **sage:\*default-document-type\*** for your site to your book design.

**sage:define-book-design** *name options &body forms* *Function*  
 A **sage:define-book-design** form can contain six kinds of forms: **use**, **define**, **modify**, **first**, **counters**, and **collectors**.

- use** Takes a book design and inherits the attributes of that design. Generally you want to **use** Generic-Generic and one or more other predefined book designs. If you want to supersede some of the predefined internal book designs (see the section "Book Design Elements", page 366), you might find it easier to specify explicitly which internal book designs you want, omitting Generic-Generic. The Lisp command **sage:graph-book-design-users** is useful to check which book designs use which other book designs so that you can be sure you are omitting those you do not want included. See the function **sage:graph-book-design-users**, page 426. **sage:graph-book-design-uses** can also be used to see exactly which book designs are being included in any given book design.
- define** Redefines some attribute(s) of the parent book designs completely or defines some environments which are exclusive to *name*. You specify all the parameters for the attribute(s).
- modify** Modifies some attribute(s) of the parent book designs. You specify only those parameters you want to change; all others are inherited from whatever the parent design is.
- first** Sets up the top-level environment (**envr**), which defines the paper size and margins, and the environment (**init**) which calls the initialize function for the book design. The initialize form can set up page headings and footings.
- collectors** Lets you define behavior associated with the frontmatter and endmatter of a book.

Its syntax is

```
(define-book-design ()
  ...
  (collectors
   collector-spec-1
   collector-spec-2
   ...))
```

Each *collector-spec-n* looks like

```
(collector-name &key name init)
```

*collector-name* is a name from the set {Contents, Tables, Figures, Index}

*name* should either be a string or a symbol. If a string, it is presumed to be the "name" for that document sec-



tion, and will be emitted as Prefatory text for that section. For example, the default name for Contents is "Table of Contents". If a symbol, it should name a function of no arguments that is called within the context of **sage:making-sage-directives** to produce the name for the section (see **sage:making-sage-directives**). *name* is not required. If not specified, no name appears.

*init*, if present, should be a symbol naming a function of no arguments that is called within **sage::making-sage-directives** to initialize formatter variables, etc. before this part of the document is formatted. It can change the pageheadings, change the rendering of the page counter, etc. Often it will do **NewPage UntilEven**. The default *init* for Contents is **sage::init-standard-table-of-contents**. The default *init* for Index is **sage::init-standard-index**.

Please note: The **collectors** clause lets you define behavior, not modify it. Therefore, if you want the table of contents to come out as "ToC", you need to include:

```
(contents :name "ToC"
          :init init-standard-table-of-contents)
```

## counters

Lets you define behavior associated with section headings like chapter, subsection, and the like. For example, in the book design element **sage::numbered-sections**, a counter for Chapter is defined as follows:

```
(chapter
  :numbered "@1."
  :referenced "@1"
  :title ((title counter-contents)
          (idirs
            (envr (HD1 above 2 below 2 facecode b)
                  counter-contents (spaces 2) title)))
  :contents ((title counter-contents)
             (render-regular-toc
              '(Tc1) title counter-contents)))
```

In this example, the keywords specify the following:

**:numbered** The numbering template used in the chapter opening. For examples of numbering templates, see the section "Defining Counters for an Article", page 405.

<b>:referenced</b>	The numbering template used in references to the chapter, for instance in the table of contents.
<b>:title</b>	The formatting specifications for the chapter opening. This is where the environment <code>Hd1</code> is used.
<b>:contents</b>	The formatting specifications for the table of contents for chapter entries. This is where the environment <code>Tc1</code> is used.

**sage:define-box-type** *box-type-name document-type device-type* *Function*  
*&rest options &key :left :top :right :bottom :horizontal :vertical :all*

Defines boxes to be used for boxing text in book designs.

**sage:define-crossreference-appearance** *appearance-name documentation booleans &key :formatter :editor* *Macro*

Defines the appearance of a crossreference link type.

*appearance-name* The name for this appearance. It should be a symbol. This name (string-capitalized) will appear in appearance name menu you get when editing a link.

*documentation* A string providing brief documentation for this link. It will appear when the mouse is moved over this link type in the menu.

*booleans* A set of options to this link type. Each option is of the form *option* or (*option default &key :documentation :pretty-name*)

*option* is a symbol naming the option.

*default* is the default value. Since these are booleans, the default must be **t** or **nil**.

*:documentation* is a string providing brief documentation of the purpose of this option. It will appear when the mouse is moved over this option in the menu.

*:pretty-name* is a string which appears in place of this option name in the menu.

**:formatter** A form whose **car** is an argument list and **cdr** is a functional body. The body can reference the variables in the argument list, as well as any of the booleans. The two arguments passed to this function are a record-group and a "contents-list".

So it is ((*record-group contents-list*) &body *body*)

The *record-group* is the particular record-group this link points to. The *contents-list* is a sage data structure which corresponds to the formatted name of that *record-group*.

Although **:formatter** is a keyword argument, it is required.

The formatter should do output to **\*standard-output\***. It is called within the scope of a **sage:sage-formatting**, so it can use things like **sage:sage-paragraph** or **sage:in-environment**. If it wants to output the formatted name of the target of the link, it can use (**sage:sage-contents-list t contents-list**).

The output from formatter is what appears in place of the link when the sage formatter runs. So the formatter for the See appearance outputs things like "See the ..."

#### **:editor**

A form whose **car** is an arglist and **cdr** is a functional body. The body can reference the variables in the arglist, as well as any of the booleans. The two arguments passed to this function are a record-group and a stream.

So it is ((*record-group stream*) &body *body*)

The editor should do output to stream. This is how the link displays itself in the editor.

If there is no editor, the formatter is run and its output is what shows up in the editor.

<b>sage::define-device-type</b> <i>name</i>	<i>Function</i>
Defines a device type.	
<b>sage::define-document-type</b> <i>name</i>	<i>Function</i>
Defines a document type.	
<b>sage:define-line-type</b> <i>line-type-name document-type device-type</i> &rest <i>options</i> &key <i>:weight :groove :halftone</i> <i>:character :rawfont</i>	<i>Function</i>
Defines a line for use in highlighting text in book designs.	
<b>sage:define-sage-attribute</b> <i>name argument-type environment-flavor form</i>	<i>Function</i>
Defines a new environment attribute.	
<b>sage:define-sage-command</b> <i>name arglist</i> &key <i>:document-type :device-type :definition :text? :starts-with-printing-char? :ends-with-newline? :ends-with-space?</i>	<i>Function</i>

Defines a command to be used in a .sab file. Usually you do this in the course of creating a specialized document type. See the section "Creating a Letter Document Type", page 393.

**sage::define-sage-counter** *name document-type device-type definition* *Function*

Defines a new counter type.

**sage::define-sage-font** *name document-type device-type definition* *Function*

Specifies a font to be used with the formatter for hardcopy output.

**sage::describe-environment** *envr* *Function*

Takes an formatting environment and displays its parameters and attributes. For example:

```
(sage::describe-environment 'sage::description)
The definition of SAGE::DESCRIPTION in
#<Directive ENVIRONMENT DESCRIPTION> looks like this:
(BREAK :NONE)
(SAGE::CONTINUE :NONE)
(SAGE::ABOVE (1 SAGE::LINES))
(SAGE::BELOW (1 SAGE::LINES))
(FILL :NONE)
(SAGE::LEFTMARGIN (+ 16 SAGE::CHARACTERS))
(SAGE::SPACES SAGE::COMPACT)
(SAGE::INDENT (- 16 SAGE::CHARACTERS))
(SAGE::SPACING (1 SAGE::LINES))
(SAGE::UNNUMBERED :NONE)
(SAGE::PARAGRAPHBREAKS SAGE::NORMAL)
```

**sage:graph-book-design-users** *name* *Function*

Draws a graph of all the document designs that use the document design *name*.

```
Command: (sage:graph-book-design-users 'sage::3symanual-lgp2)
LISP-DICTIONARY-LGP2
ARTICLE-LGP2
3SYMANUAL-LGP2
NIL
Command:
```

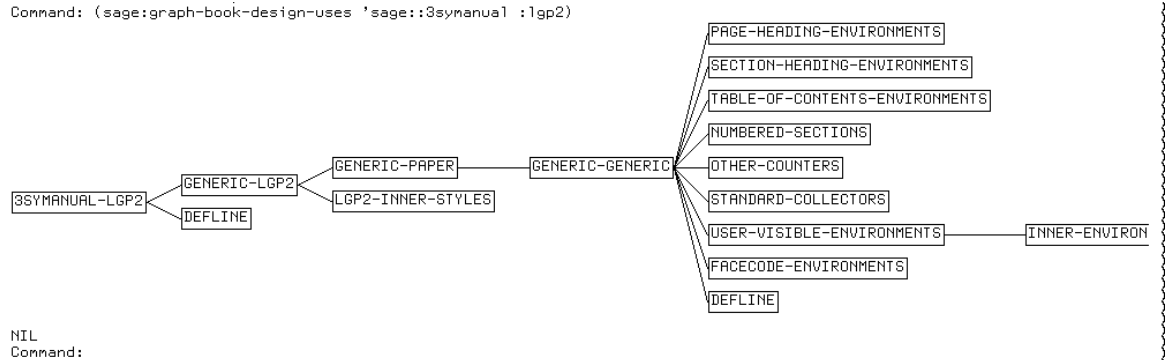
```

graph LR
    A[LISP-DICTIONARY-LGP2] --> C[3SYMANUAL-LGP2]
    B[ARTICLE-LGP2] --> C
  
```

Figure 70. **sage:graph-book-design-users**

**sage:graph-book-design-uses** *document-type device-type**Function*

Draws a graph of the document designs from which the document design *document-type device-type* inherits.

Figure 71. **sage:graph-book-design-uses****sage:idirs** &body *body**Macro*

Allows you to insert formatting directives for the formatting of titles in counter definitions, and for the formatting of counter contents for tables of contents, indexes, and pageheadings.

spaces	Accepts a number and puts in that many literal spaces. It uses the command <b>sage::literal-space</b> .
L	Accepts a text string and puts it in the LS character style, Lisp like typeface.
R	Accepts a text string and puts it in the roman character style.
envr	Accepts a sage environment and any necessary set up for placement of generated text. It uses the form <b>sage:in-environment</b> .
command	Accepts a sage command and evaluates it.
value	Accepts a counter or string with a value and retrieves the value.
lisp	Accepts a lisp form and executes it as a <b>progn</b> .

Here is a chapter counter, using **sage:idirs** to control how its title appears.

```
(chapter
  :within majorpart
  :numbered "@1."
  :referenced "@1"
  :title ((title counter-contents)
    (idirs
      (envr (HD1 above 2 below 2 facecode b)
        counter-contents (spaces 2) title)))
  :contents ((title counter-contents)
    (render-regular-toc '(Tc1) title counter-contents)))
```

**sage:\*lgp2-default-character-style\*** *Variable*  
 Specifies the character styles to be used with a particular document type.

**sage:making-sage-directives** (&optional *stream*) &body *body* *Macro*  
 Binds *stream* to a special stream, which collects all formatting and output done in *...body...*, making Sage directive structure out of them, which it returns. It must run in a **sage:with-sage-device** dynamic environment.

**sage:note-book-design-specifics** *document-type device-type entry-name* *Function*  
 Associates a document type and a device type to produce a book design.

For example, 3symanual-lgp2 is produced by the form

```
(note-book-design-specifics '3symanual 'lgp2 '3symanual-lgp2)
```

**sage::register-book** *topic-string &key (:mnemonic "") :document-type :highest-structural-level :cover :remarks :sym-copy :doc# :effectivedate :releaseversion :marketing :mitcopy :authorgroup :design :cover-printer :text-printer :printer :confidential :doctrademarks :deferred-home :sage-variables* *Function*

Registers a Symbolics Concordia topic as a book and specifies its book design.

**:document-type** {**sage::3symanual**, **sage::approach**, **sage::reference-cards**, **sage::lisp-dictionary**} Specifies the document type.

**sage::3symanual** - The standard format of the Symbolics Document Set.

**sage::approach** - The new *Approachability* book design.

**sage::reference-cards** - The smaller format for the Symbolics Reference Cards.

**sage::lisp-dictionary** - The format for dictionaries, for example books 2B and 7B in the Symbolics Document Set.

**:highest-structural-level**

Tells the formatter the overall structure of the book. The default is to have Majorparts, subdivided into chapters (like *Symbolics Concordia* , for example). To specify that book is only one structural unit, use **:highest-structural-level 'sage:chapter**.

**:sage-variables**

Allows you to set default document formatting directives. This is a list of variable/value pairs that sets Case selector variables used in the book. Using this option to **sage::register-book** is equivalent to using the Set Sage Variables command for each variable/value pair in the list.

For example,

```
(sage::register-book "My Book"
  :document-type 'sage::user-guide
  :sage-variables '((Country "USA")(Version "4.0")))
```

"My Book" uses the Case selectors, Country and Version. By default Country is set to USA, and Version is set to 4.0. Note that you can override these document formatting directives when you format the book (use the **:query** keyword of the "Format Pages Page Previewer Command ").

**43.12.5. Sectioning Commands for Conversion Compatibility****43.12.5.1. AppendixSection Command**

Starts an Appendix section. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup.

**43.12.5.2. Chapter Command**

Starts a new chapter. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup.

**43.12.5.3. MajorPart Command**

Starts a majorpart. This command is retained for conversion compatibility. In general each part of a document should be a record to provide good modularity for Document Examiner lookup.

**43.12.5.4. Section Command**

Starts a new section in the document. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup.

**43.12.5.5. SectionRef Command**

Retrieves and prints the value associated with the section counter. This command is retained for conversion compatibility. In general each section should be a record to provide good modularity for Document Examiner lookup and references are done using cross reference links.

**43.12.5.6. SubSection Command**

Starts a new subsection in the document. This command is retained for conversion compatibility.

**43.12.5.7. SubSubSection Command**

Starts a new subsubsection in the document. This command is retained for conversion compatibility.



## 44. Guide to Documentation System Maintenance

### 44.1. Setting up Your Documentation System

A *system* is any program that has grown too large to be contained in one file and has been split into a set of files that together form the working program.

Your documentation system is the set of files that hold the records that makes up your online and hardcopy documentation. A documentation system can be a stand alone system, like the Symbolics Documentation System, or it can be a *component system* of another system, like the documentation for Symbolics layered products such as Fortran or C. This means that when the C system is loaded into a Genera world, one component of the C system is its documentation, which is loaded into Document Examiner ready for lookup.

In order for the various files that make up a system to work together, they must be connected so that functions or routines in one file can communicate with other routines in another file; or, in the case of documentation, so that the records that make up a long document can reside in several files of manageable size. The links and relationships that make this communication possible are set up by the *compiler* using the Symbolics System Construction Tool. (For more detail, see the section "Introduction to the System Construction Tool" in *Program Development Utilities*.)

#### 44.1.1. Compiling

*Compiling* is the process of running the compiler program over the list of files in the *system declaration* to produce the list of binary files that get read into Document Examiner.

For the Symbolics Documentation System, this is done in parallel with recompilations of other systems that make up our Lisp World so that World Building (the process of loading all the systems into core memory and then saving an executable image of that memory) can be done as economically as possible. In the case of a documentation system that is a component of another system, compilations should be coordinated with the developers of the parent system, since the recompilation of the parent system includes loading the component systems in the same way that building a world loads the Symbolics Documentation System.

#### 44.1.2. The System Declaration for a Documentation System

A *system declaration* is a Lisp form that provides the information necessary for the compiler to construct a system from its various files. It is in a Lisp file usually named *system.lisp* that is stored in the directory for the system.

Here is a sample system declaration for a small documentation system. Its pathname is `sys:sched;doc:sched-doc.lisp`.

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER; Base: 10 -*-
```

```

(sage:define-documentation-system sched-doc
  (:pretty-name "Scheduling and Planning Documentation"
   :default-pathname "sys:sched;doc;"
   :patchable t
   :advertised-in (:herald)
   :maintaining-sites (:down-under)
   :distribute-sources :always
   :distribute-binaries :always
   :default-module-type :sage
  )

  (:module schedule ("scheduler" "calendar" "notify"))
  (:module plan ("long" "short"))
  (:module installation ("install"))
  (:module illustrations ("sched-pictures.pic")
   (:type :data-no-load)
   (:source-category :restricted))
)

(sage::register-book "Scheduling and Planning Made Easy" "")
(sage::register-book "Scheduling and Planning System Installation Guide" "")

```

The parts of this form, sometimes referred to as a *sysdcl* (pronounced *sys-dicle* to rhyme with popsicle) are:

### **sage:define-documentation-system**

The Lisp defining form that defines a system.

**sched-doc**           The short form of the name, used in commands such as Load System.

### **:pretty-name** option for **defsysteem**

Specifies the name of the system for use in printing.

### **:default-pathname** option for **defsysteem**

Specifies a default pathname against which all other pathnames in the system are merged.

### **:patchable** option for **defsysteem**

Specifies whether you want the system to be patchable or not.

You want to have this option set to **t** for Yes so that you can make incremental updates.

### **:advertised-in** option for **defsysteem**

Specifies the contexts in which the system name and version number should be displayed.

The default is (**:herald**). Additional possibilities are **:finger** and **:disk-label**. For example:

```

:advertised-in (:herald :disk-label)

```

If you do not want the presence of the system advertised, even in the herald, specify:

```
:advertised-in ()
```

**:default-module-type** option for **defsystem**

Specifies a keyword, which is the default type for each module. You want to specify **:sage**. This tells the compiler how to compile your system modules.

**:maintaining-sites** option for **defsystem**

Specifies the list of sites that maintain the system. This determines who can patch the system. You want to specify your site.

**:distribute-sources** option for **defsystem**

Specifies whether or not the sources are distributed.

Although .sab files are binary files, they are source files as far as Symbolics Concordia is concerned. You want to specify **t** for **:distribute-sources**.

**:distribute-binaries** option for **defsystem**

Specifies whether or not the binary files (object files) for the system are distributed.

There are no binary files for Symbolics Concordia in the sense of a file of type .bin, so you want to specify **nil** for this.

**:module**

One or more named modules that make up the body of the documentation. Each module specifies a list of files that contain the actual documentation records. For example, the module `plan` contains two files, `sys:sched;doc;long.sab` and `sys:sched;doc;short.sab`. Usually you create modules based on the organization of the documentation, but the location of a particular file or record in a module does not limit its use in another part of the documentation.

If you have pictures in your documentation (see the section "Inserting Illustrations in Your Symbolics Concordia Documents", page 179) you should have a module like the module `illustrations` in which to list the source files for the pictures. This module should be of type **:data-no-load**, meaning that the compiler and loader ignore it, it is just there for book-keeping purposes. You probably want to make this module **:source-category :restricted**, also, since pictures can only be edited by someone who has the Graphic Editor software.

Additional options that you might want or need include:

**:default-package** option for **defsystem**

Specifies the name of an existing package into which each file in the system will be loaded or compiled. This is particularly important if your documentation system is documenting a lan-

guage system that uses its own packages. It is very important that your documentation system use the same packages as the software it documents. See the section "System Status in the Herald", page 445.

**:initial-status** option for **defsystem**

Sets the initial status of the system when a new major version is created. The possible values for status are **:experimental**, **:released**, **:obsolete**, and **:broken**. The default is **:experimental**. See the section "Releasing a Documentation System".

**:journal-directory** option for **defsystem**

Specifies the location of the *journal* directory. The journal directory is where the list of files in each system version and all the patch files are stored. The default is a subdirectory called *patch*; under the default pathname. If you want the journal directory for your documentation system to go in a different place, for example, with the journal directory for the rest of the software system it documents, you would use this keyword to specify that directory.

For example, the journal directory for *sched-doc* is `SYS:SCHED;SCHED-DOC;PATCH;.` The journal directory for the *sched* system as a whole is `SYS:SCHED;PATCH;.` To put the *sched-doc* journal files in the same directory as the *sched* system as a whole, you specify:

```
:journal-directory "sys:sched;patch;"
```

See the section "The Journal Directories", page 450.

**:bug-reports** option for **defsystem**

Specifies the mailing list for bug reports and the purpose of the bug mail. For example:

```
:bug-reports (:name "sched-doc"
              :mailing-list "bug-sched-doc"
              :documentation "Comments on the Scheduler Documentation")
```

After the **sage:define-documentation-system** form you want to have one or more **sage::register-book** forms. See the function **sage::register-book**, page 428. If your documentation system contains more than a couple of books, it is a good idea to separate the **sage::register-book** forms out into their own file. See the section "A System Declaration for a Large Document Set", page 436. If your documentation system is a component of another system, you should check with your system developers to make sure that it is included in that system declaration.

For complete information on system declarations: See the function **defsystem** in *Program Development Utilities*.

### 44.1.3. Registering Your Documentation System

When you use a command such as Load System on a system that is not loaded and thus not defined in your world, the System Construction Tool looks in the directory SYS:SITE; for a file named *system-name*.SYSTEM, which contains the information to locate the system directory and the system declaration.

You should create such a file for your documentation system. You use the form **sct:set-system-source-file** to indicate the name of the system declaration file. The file SYS:SITE;SCHED-DOC.SYSTEM might look like this:

```
;;; -*- Mode: LISP; Syntax: Common-Lisp; Base: 10; Package: SCT -*-
```

```
(set-system-source-file "sched-doc" "sys:sched;doc;sched-doc")
```

This assumes that there is already a logical pathname "sys:sched;" set up. If not, then **fs:make-logical-pathname-host** should be used to set it up:

```
;;; -*- Mode: LISP; Syntax: Common-Lisp; Base: 10; Package: SCT -*-
```

```
(fs:make-logical-pathname-host "sched")
```

```
(set-system-source-file "sched-doc" "sys:sched;doc;sched-doc")
```

For more details on logical hosts and system source files:

See the function **fs:make-logical-pathname-host** in *Program Development Utilities*.

See the function **sct:set-system-source-file** in *Program Development Utilities*.

### 44.1.4. Updating the System Declaration

Each time you add a file to your documentation system, you should add the file name to the appropriate module in your system declaration.

If your system declaration gets updated frequently between compilations, or if it is being updated by many people, it is often better not to touch the actual system declaration file but rather to have a copy of that file, perhaps with *new-* prepended to the file name, to which updates are made. Then when you are ready to compile the system, you check the *new-system.lisp* file and copy it to *system.lisp* just before compiling. By doing this you insure that the copy of *system.lisp* always corresponds exactly to the compiled version of the system in the world.

### 44.1.5. Packages for Documentation Systems

If your documentation system is not a stand alone system using the standard Genera packages and Lisp syntax, for example, it is for another software system, such as Fortran or Prolog, or your own application product, you must be sure that your environment is set up to use the correct packages and syntax. For a description of the Lisp package system and how it works: See the section "Overview of Packages" in *Symbolics Common Lisp Language Concepts*.

When you are documenting language objects, Symbolics Concordia interacts with the software environment to provide consistency checks between the language ob-

jects records you create and the software. Whenever you create a language object record, Symbolics Concordia checks that an object of that name exists in the world and supplies the argument list automatically for you. If you do not have the software you are documenting loaded, the objects you try to create records for do not exist in the world. In this case, Symbolics Concordia creates the record for you anyway but warns you that no such object exists. The record that Symbolics Concordia has created is a *phony* language object record, and even after you load the software, that record has no relationship with the language object in the world. This can cause unexpected errors. *Always have the software system you are documenting loaded before you start working on the documentation system.*

For some systems, the *dependencies* of one module upon another make the order in which modules and other systems are loaded important. In these cases it is particularly important to make sure that everything loads in the right order without errors. Continuing through load errors can result in inconsistencies in your world. Symbolics Concordia allows you to continue through certain errors for those cases where the mismatch of records is a writer error that you want to fix; but it is you, the writer, who must understand where the error is coming from. So, before continuing through any consistency errors Symbolics Concordia signals, be certain that your environment is properly set up with the right software packages loaded in the right order.

One way to insure that you always have the correct software loaded is to coordinate with the developers of the software you are documenting and use the same world versions that they are running.

If you want your documentation system to be separately loadable from your software, perhaps so someone can peruse the documentation to decide if they want to load the software, you should make sure that the packages used by your software are loaded as part of your documentation system. One way to do this is to have a separate system that sets up the packages for the software. This system can then be loaded as a component of both the software system and your documentation system. Your software developers can set up such a system and then you can declare the package system as a module of **:type system** in your documentation system declaration. See the section "The System Declaration for a Documentation System", page 431.

#### 44.1.6. A System Declaration for a Large Document Set

When your documentation system contains many books, it is a good idea to remove the **sage::register-book** forms from the system declaration file and place them in a *book registry* file by themselves. Here is an example of a book registry file:

```

;;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER; Base: 10 -*-
;;;
;;; The following sets up the book registry for the
;;; Scheduling Documentation
(sage::register-book "Scheduling and Planning Made Easy"
                    :document-type 'approach
                    :highest-structural-level 'chapter
                    :Confidential "Company Confidential"
                    :mnemonic ""
                    :Doc# "100051"
                    :Marketing "No"
                    :Remarks "")
(sage::register-book
  "Scheduling and Planning System Installation Guide"
  :document-type '3symanual
  :highest-structural-level 'chapter
  :Confidential "Company Confidential"
  :mnemonic ""
  :Doc# "100050"
  :Marketing "No"
  :Remarks "")
.
.
.

```

The book registry file is then made into a module of the documentation system. Here is how the system declaration file is modified to load a book registry. The changed lines are marked by a vertical bar (|).

```

;;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER; Base: 10 -*-

```

```

(sage:define-documentation-system sched-doc
  (:pretty-name "Scheduling and Planning Documentation"
   :default-pathname "sys:sched;doc;"
   :patchable t
   :advertised-in (:herald)
   :maintaining-sites (:down-under)
   :distribute-sources nil
   :distribute-binaries t
   :default-module-type :sage
   :source-category :optional
  )
| (:module books "book-registry.lisp" (:type :lisp))
  (:module schedule ("scheduler" "calendar" "notify"))
  (:module plan ("long" "short"))
  (:module installation ("install"))
  .
  .
  .
| (:serial books
  |      (:parallel schedule plan installation ...))
)

```

Notice especially that a new option, **:serial** has been added. This declares the dependencies for loading: the **sage::register-book** forms must be loaded before the other modules, but the order of loading the other modules is not important. Hence the **sage::register-book** and *other-modules* have **:serial** dependency, and the other modules have **:parallel** dependency among themselves. **When you add a new module to the system you must remember to add it to this form at the end of the file or else it will not get loaded.**

## 44.2. Compiling a Documentation System

There are three main parts to compiling a new documentation system:

- "Making Sure the System Declaration is Correct"
- "The Actual Compilation"
- "Reap Protecting the New Version"

Additionally, you should compile the system for all the machine types (3600-family and Ivory, for example) on which the system is to be loaded.



### 44.2.1. Making Sure the System Declaration is Correct

Check your system declaration to make sure that the "right" things are going to get included. For example:

- Do all the **sage::register-book** forms look correct? Have any books been added that are not listed there? Have any titles changed?
- Are all the modules listed in the **defsystem** included in the *dependencies* (**:serial** and **:parallel**) at the end of the form?
- Is anything commented out that should be included? Is anything included that should be commented out?

When you are satisfied that everything is okay, you are ready to proceed. If you are maintaining two copies of your system declaration, *new-system.lisp* is the file you check and now is when you copy it to *system.lisp*.

### 44.2.2. The Actual Compilation

Turn off More processing with `FUNCTION-Ø-M`.

Consider how you want redefinition warnings to be handled. The variable **sage:\*inhibit-redefinition-warnings\*** controls the warnings. The default is **:just-warn**. If you are recompiling in a world with your documentation system already loaded, the loading operation of the compiler may be interrupted by warnings if there are many patches. For each record that has been patched, it prints out a warning that the record "... was previously defined in *some-patch-file-or-other*."

The redefinition warnings are important to locate real duplicate definitions, so you should at least scan them at the end of the process to make sure they are all just for patches.

If you prefer, you can have the compilation stop at each such warning, which gives you the opportunity to check right then and there that it is ok, then allow the compilation to proceed. You do this by typing the form

```
(setq sage:*inhibit-redefinition-warnings* t)
```

to your Lisp Listener. However, you then have to watch the compilation process or else it ends up sitting for a long period of time asking "Is this ok? (Y or N)".

Of course, the best world to recompile your documentation system in is a specially built world that has your software system loaded but not the documentation. Sometimes such a world is constructed, so check with your developers to see if there is an appropriate one available. If you can use such a world (it must have **nsage** and **Symbolics Concordia** loaded, plus the appropriate software system, but not your documentation system), the only redefinitions possible are those arising from actual duplicate definitions found by the recompilation.

Compile the documentation system by doing:

```
Compile System your-documentation-system
```

This reads the system declaration file, creates the journal file consisting of the .newest version of each .sab file, tells you what documentation system version has been created, and finally loads all the files. How long this takes depends on how many files need to be loaded. For the Symbolics Documentation System, the loading process takes about one hour.

In the event of **SCT** bugs involving the system declaration file, make sure the system declaration is correct, parse it by hand (with `c-sh-E`) and do:

```
(compile-system 'your-doc-system :no-reload-system-declaration t)
```

#### 44.2.3. Reap Protecting the New Version

When you have finished, **DO NOT FORGET** to reap protect the files of the system version you have just compiled!

You do this with **sct:reap-protect-system**

```
(sct:reap-protect-system "your-doc-system" :version n)
```

It is a good idea to look at a the directory to make sure the *dont-reap* bit (\$) appears next to the most recent version of each .sab file in the system.

At this point you can tell your developers that the new documentation system is ready for loading.

#### 44.2.4. Compiling a Documentation System for Multiple Machine Types

If you have both Ivory and 3600-family machines, or if you are going to distribute your documentation system to both types of machines, you must repeat the compilation step on the other machine type. Since you do not want to create a new version of the system, you use the `:Version` keyword:

```
Compile System your-documentation-system :version n
```

*n* is the version number you just created. For more detail, see the section "Compiling a System for Multiple Machine Types" in *Program Development Utilities*.

If your system is very large, or otherwise difficult to compile so that doing it twice is too much overhead, there is a way to "compile" a documentation system for the other machine type by hand. Documentation systems consist of .sab files, not .bin or .ibin files. .sab files can be loaded by both machine types. So, compiling for the other machine type really is only a matter of updating the journal files to record both machine types. You can edit the `.component-dir` for the version of your system that you just created and add the other machine type.

For example, here is the `.component-dir` for a documentation system that was compiled on an Ivory machine:

```
(( "DOC" 424)
;; Files for version 424:
(:IMACH
 (:DEFSYSTEM
  ("SYS:DOC;DOC.LISP" 223 NIL))
 (:INPUTS-AND-OUTPUTS
  ("SYS:DOC;DEFBOOKS.LISP" 90 NIL)
  ("SYS:DOC;ANSI-CL;ANSI1.SAB" 8 NIL)
  .
  .
  .
  ("SYS:DOC;ZMAILT;ZMAILT4.SAB" 28 NIL)
  ("SYS:DOC;ZMAILT;ZMAILT5.SAB" 17 NIL))))
```

To "compile" this for 3600-family machines, simply copy the (:IMACH ...) form, placing the copy just after the (:IMACH ...) form. Change the :IMACH in one of the forms to :|3600|. The result should look like this:

```
(( "DOC" 424)
;; Files for version 424:
(:IMACH
 (:DEFSYSTEM
  ("SYS:DOC;DOC.LISP" 223 NIL))
 (:INPUTS-AND-OUTPUTS
  ("SYS:DOC;DEFBOOKS.LISP" 90 NIL)
  ("SYS:DOC;ANSI-CL;ANSI1.SAB" 8 NIL)
  .
  .
  .
  ("SYS:DOC;ZMAILT;ZMAILT4.SAB" 28 NIL)
  ("SYS:DOC;ZMAILT;ZMAILT5.SAB" 17 NIL)))
(:|3600|
 (:DEFSYSTEM
  ("SYS:DOC;DOC.LISP" 223 NIL))
 (:INPUTS-AND-OUTPUTS
  ("SYS:DOC;DEFBOOKS.LISP" 90 NIL)
  ("SYS:DOC;ANSI-CL;ANSI1.SAB" 8 NIL)
  .
  .
  .
  ("SYS:DOC;ZMAILT;ZMAILT4.SAB" 28 NIL)
  ("SYS:DOC;ZMAILT;ZMAILT5.SAB" 17 NIL))))
```

Write the .component-dir file out and the system can then be loaded on either machine type.

#### 44.2.5. Summary of Compiling the Documentation Database

Here are the steps in the compilation process again:

1. Check the system declaration file.
2. Turn off More processing: `FUNCTION 0 M`
3. (If necessary) **(`setq sage::*inhibit-redefinition-warnings* :just-warn`)**
4. Compile: `Compile System your-documentation-system`
5. Reap Protect the result: **(`sct:reap-protect-system "your-documentation-system" :version n`)**

#### 44.2.6. Patching a Documentation System

Once a world is built with a version of a system in it, or a system is compiled with a component documentation system loaded, any changes made to the sources of that system (editing or adding documentation in the case of a documentation system) are not reflected in the world or the parent system. However, if you recompile the system to pick up the changes, the newly recompiled system cannot be loaded into the same version of the world (or included in the current version of its parent system) because there is already a copy of that system there. A new world must be built or the parent system must be recompiled to incorporate the newly recompiled system. Obviously, you want a way to incorporate changes short of re-compilation. This is accomplished by *patching*.

*Patching* is the process of adding incremental changes to a system.

The procedure is as follows:

1. In a Lisp Listener: Load Patches
2. `SELECT W`
3. `m-X Start Patch`
4. It prompts: System to patch: *your-documentation-system*
5. You can patch records individually by locating them with `s-`, and then using `m-X Add Patch`.

Alternatively, you can use `m-X Add Patch Changed` records, which locates each changed record and queries you whether you want to include it in the patch.

You can repeat either of these last two steps as many times as you like.

6. `m-X Finish Patch`

## 7. *m-X* Load Patches *your-documentation-system*

### 44.2.6.1. Why Patching Documentation is Important

Patching is very important in the Symbolics Concordia environment. It ensures that the record registry is always up to date, greatly reducing the possibilities of accidentally creating two records with the same name.

It allows incremental online review, since your developers and other interested people can load documentation patches and offer you feedback.

It also facilitates coordination among a large group of writers since each can keep up with what others are doing and not duplicate effort.

### 44.2.6.2. Strategies for Patching Documentation

Ideally, all changes made to a document that is part of a system should be patched into the system. However, sometimes there are reasons to put off patching some work into a system.

- New work that constitutes an entire new section of documentation. You probably want to wait until you feel comfortable with the quality of the new documentation.
- Large-scale reorganizations. You probably want to leave a consistent online version of the documentation until you have completed at least the structural reorganization, if not the complete update.

These are both valid reasons for wanting to avoid patching. However, the importance of the incremental aspects of using Symbolics Concordia cannot be overemphasized.

If two writers both have new work that is not part of the system, they can inadvertently create records with the same name. In addition, they miss out on the opportunity to share information and might duplicate each other's work instead of being able to reuse each other's records. It is far better to patch in sections of a developing document as they are worked on, perhaps indicating in the patch comment that the work is a preliminary draft.

In large-scale reorganizations, if you do not scrupulously read in all the files for the document you are reorganizing, you can confuse the record registry in your machine about renamed records and rearranged links. The structural reorganization is so easy to do that it makes more sense to patch it in as soon as it is stable and then work incrementally on sections of the reorganized document, patching as each section is updated.

There is one situation when you should not patch: when you have killed a record that was never patched or compiled into a documentation system. If you have just created a record and then decide that you do not need it and kill it, you do not need to patch the killed record. If you created this short lived record in a file that belongs to a documentation system, Add Patch Changed Records offers to undocument it for you. You should press *N* at this point, and decline the offer. Since the

record never existed in anyone else's environment, patching it would cause an error when the patch is loaded.

#### 44.2.6.3. When You Cannot Patch

After a documentation system has been recompiled but before this new system has been included in a world, you do not want to patch the old system, since it is now obsolete, and you do not have access to the new system so you cannot patch that. This situation is referred to as a *Patch Freeze*. If you have coordinated closely with your developers or world builders, this freeze should be brief and you might choose to simply suspend editing of .sab files until the freeze is over. If you do make changes to your .sab files during this period, you should keep track in some way of the records you change so that you can patch the changes into the new system when it is available.

"Add Patch Changed Records" keeps track of the records you have modified but not patched in your current boot session, but since you will probably have to boot to get your new documentation system, this does not help in this case. Using "List Changed Records" and saving the result in a file or on paper is a good way to keep track of modifications.

Killing records presents another problem. If you save out the file from which you have killed a record, the marker indicating that a record has been killed is removed from the file and you then are unable to patch the killing of the record later. If you kill records when you are unable to patch, you will have to wait until the next recompilation to actually have the records removed from the record registry. One way to get around this problem is to remove all the links from the record to be killed and move it to a file by itself or with other records to be killed. Later when you can patch, you can kill all these records at once and patch them out of the record registry.

#### 44.2.6.4. Working on a Document Outside Its Documentation System

In "Why Patching Documentation is Important" we mention some of the good reasons to patch documentation systems incrementally, even for work in progress. However, writers are usually loath to have their half-baked drafts in public view.

There is a danger with Symbolics Concordia: we do not recommend developing documentation outside of a documentation system, but you can work that way if you follow a few simple rules.

1. Work modularly so that you are adding small finished pieces of a document to the system at intervals.
2. Remember that other writers with whom you may be cooperating do not have access to any new record names you use. (Similarly, you do not have access to any of their new names, if they are also working outside the system.) You must take care of checking for name conflicts with any of those private records yourself (the Record Registry takes care of checking for conflicts with published records that are patched into the system, of course). One way you might avoid conflicts is to use a prefix, perhaps your login id, in each new

record you create (see "Set Record Name Prefix Command" and "Set Record Name Suffix Command").

3. Avoid making links to the documentation system to avoid problems with records that might be undocumented before you finish your work. You might flag locations in your text where you want links to other records in the system with a character string you can search for ( `&$*`, for example), and a parenthetical comment about what kind of link and its intended target.
4. Always have all the documentation you are working on outside the system loaded into your environment, so you do not create name conflicts within your document. You can use Load File on a `.SAB` file to load its index information into the Record Registry in your machine if you do not want to read in the file itself.

When you were ready to finally release your documentation,

1. Load all patches for your documentation system.
2. Rename those records to the correct names.
3. Make links at any locations you flagged. Making your document a Tag Table allows you to use Tag Search to locate all the strings you left to indicate links. See the section "Tag Tables and Search Domains in Zmacs" in *Editing and Mail*.
4. Patch your new documentation into the system. (See the section "Patching a Documentation System", page 442.)

### 44.3. Setting a System's Status

There are two types of *status* a system can have.

- The status as displayed in the system herald, usually "Experimental *system-name*" or just plain *system-name*. The latter indicates that the system has been *released*.
- The status for purposes of loading (or other commands that operate on systems such as Show System Components). This is recorded in the journal files. (See the section "The Journal Directories", page 450.)

#### 44.3.1. System Status in the Herald

A system name is displayed in the herald as "Experimental *system-name*" by default. If you want it to display as just *system-name*, you use the form **set:release-system** after you have compiled the system. For example, to release version 3 of `sched-doc`, you would type this form at a Lisp Listener:

```
(sct:release-system "sched-doc" 3)
```

If you have only one version of a system at a time and do not want it ever to appear in the herald as "Experimental", you can use the system declaration option **:initial-status** so that it is compiled with its status already designated. See the section "The System Declaration for a Documentation System", page 431. There are two status keywords that are useful with **sct:release-system** or the **:initial-status** option to **defsystem**:

**:experimental**      The default. It is used for developing systems and indicates that the system is changing. It displays in the herald as:  
Experimental Scheduling and Planning Documentation 3.45

**:released**            Indicates that the system is judged stable and ready for general use. It displays in the herald as:  
Scheduling and Planning Documentation 3.45  
Usually you mark a system as **:released** just before shipping it to customers.

When **sct:release-system** is used to designate a system as released, the system-dir file is updated to designate that version as **:released**.

#### 44.3.2. System Status for Loading and Other Operations

Commands such as Load System seek a version of the system that is marked **:released**. If there is no **:released** version, then the **:latest** version of the system is tried. A system is marked as **:released** by **sct:release-system**. When a new version of the system is compiled, it is marked **:latest** by default, but Load System still loads the version marked **:released** unless you designate the **:latest** explicitly in the command, or unless you release the newly compiled version. Using this mechanism, you can keep a version of your released documentation around for the use of general users and still have a development version that you load explicitly by specifying **:version :latest**.

You can add symbols to designate versions of your system for loading and other operations to accommodate special cases that might exist at your site.

## 44.4. Maintaining Your Documentation Sources

### 44.4.1. Cleaning up

Writers should, of course, be conscious of how many copies of files they are generating as they work, and clean up after themselves from time to time. However, in the absence of serious disk space problems, too rigorous cleaning up of the directories while work is going on is not necessarily good for several reasons:

1. Being able to recover a previous version of a record. Everybody makes mistakes, as proven by the statement, "I thought I made a mistake once, but I was wrong."



2. Making sure that the "history" of a file gets backed up. Dumps should be done at regular intervals, but if you are making a large number of changes and deleting extra files behind yourself, the intermediate states get deleted before they make it to a dump tape. If you find you need one of those intermediate states, you cannot recover it and must recreate it.
3. Conserving disk space. In moderation, having extra copies of files on disk actually conserves disk space. Although this appears to be a contradiction, consider what happens when a new disk is added to a system: within days (sometimes only hours) the disk space disappears. This is because files multiply to fill the space allotted to them. If you keep the directory very clean and then need to dump out a large file when your disk system is 99 percent full, you might not be able to do it unless you have something you can *delete* first. Extra versions of files are a sort of savings account of disk space. **Moderation** is a key word here, of course.

The dump bits (! indicates a file that has not yet been dumped to tape) and dont-reap bits (\$ indicates a file that should not be deleted) should always be respected when cleaning up. In a group where many writers are working in the same source directories, it is a good idea to remember the rule of thumb: "If you didn't create it, don't delete it." (The creator's login name is on the far right of the Dired display.)

Clean Directory (`m-X`) is the safest way to clear up space. If you like to "clean up after yourself", use this command on the directories you are using. Do not delete files "by hand". It is too easy to type one too many D's with dired or miss with the mouse in FSmaint. See the section "Clean Directory" in *Editing and Mail*.

See also "Clean File" .

If you do have trouble with a patch finishing due to lack of disk space, the patch directory is not the one to try to clean up. Use Clean Directory (`m-X`) on one of your source directories.

Housecleaning does need to be done occasionally, of course. Before compiling a new version of the documentation is an excellent time to clean up your directories.

#### 44.4.1.1. Deleting an Old Version of Your Documentation System

Before you delete a version of your documentation system, make sure that it is not being used. Check with whomever builds worlds at your site to find out what systems are loaded in the worlds currently in use.

Once you know that a version is not in use, proceed as follows: If you want to delete version 2 of your system, for example, you first use the function **sct:reap-protect-system** to remove the reap protect bits from the files.

The function **sct:reap-protect-system** changes the properties for all the files in a system. See the function **sct:reap-protect-system** in *Program Development Utilities*.

1. To remove the reap protect bit from all the files in version 2 of your system, you type:

```
(sct:reap-protect-system "your-system" :version 2 :reap-protect nil)
```

2. Then, and this is **VERY IMPORTANT**, you do:

```
(sct:reap-protect-system "your-system" :version 3)
```

This ensures that all the files for the next version are properly protected. This is necessary in case there are some files that are used by both version 2 and version 3 (perhaps there were no changes made to them between compilations). The reap protect bit of these files would be removed because they are used by version 2, but since version 3 uses them also, the bit should be set again.

So removing reap protection for a system is always a two step process:

1. **Remove the reap protect bit for version  $n$ .**
2. **Set the reap protect bit for version  $n+1$**

Then use Clean Directory (m-X) on your directories.

#### 44.4.1.2. Deleting Extra Versions of Files

This Situation is the one time when the rule of thumb "If you didn't create it, don't delete it" does not apply. Make a preliminary pass through your directories to check for obvious garbage. Examples of obvious garbage include (but are not limited to):

- Files of length 0.
- Files that have strange names that are clearly the result of a typing error. You can usually determine this by noticing a file of the same date and approximate time with a more consistent name. When in doubt, consult the writer who created the files.
- Very old files from obsolete documents. Once again, when in doubt consult the creator of the file.

Then do

```
m-X Clean Directory your-system-directory;*.*.*
```

See the section "Clean Directory" in *Editing and Mail*.

See also (m-X) "Clean File".

#### 44.4.2. Checking Your Documentation System

Symbolics Concordia does not permit you to create a record that has the same name as an existing record. Also, it does not permit you to create a link to a record that does not exist. However, both duplicate records and links to non-existent records can occur. For example, if two people independently create a record called "Starting Up Your Symbolics Computer" and each add this record to

a file in the same documentation system without patching, when the system is compiled there will be two records named "Starting Up Your Symbolics Computer". Similarly, if someone is working on several files, one of which contains records with links to records in another file, and the file with the linked-to records is accidentally left out of the system declaration, when the system is compiled there will be links to records that the system does not know about. Both of these problems can be avoided by careful attention to maintaining the system declaration and to patching. However, Symbolics Concordia does provide you with a way to check the integrity of your system.

When you compile your documentation system, you are automatically warned about duplicate records and about missing records, that is links that do not point at a record. You can copy these warnings to a file using Copy Output History or by marking and yanking them with the mouse. It is advisable for the person in charge of maintaining the documentation system to distribute this information to the writers so they can remedy the problems.

The other kind of problem that can arise with a documentation system is *orphan records*. An orphan record is one that is not linked to by any other record in the system, or a record that has no contents regardless of whether it is linked to by any other records. Some orphan records are intentional, but sometimes the omission of a record is an oversight. Prior to producing hardcopy books or to distributing a documentation system, it is advisable to check the system for orphan records. There are three commands to do this:

#### Find Orphan Records in Buffer

Find all orphan records in the buffer. An orphan record is one that has no links to it, or one that has no contents regardless of whether it has links.

#### Find Orphan Records in Tag Table

Find all orphan records in the tag table. An orphan record is one that has no links to it, or one that has no contents regardless of whether it has links.

You can create a tag table in several ways:

#### Select All Buffers As Tag Table

Selects all buffers currently read in. Invoked by: `m-X`  
Select All Buffers As Tag Table.

#### Select Some Buffers As Tag Table

Selects buffers currently read in, querying about each one. Invoked by: `m-X` Select Some Buffers As Tag Table.

#### Select Some Files As Tag Table

Selects some files as a tag table. Read successive pathnames from the mini-buffer. Invoked by: `m-X` Select Some Files As Tag Table.

**Select System As Tag Table**

Selects all the files in a system as a tags table. Invoked by: `m-x Select System As Tag Table`.

If you have several tag tables defined, you can select the one to use:

**Select Tag Table**

Makes a tag table current for commands like Find Orphan Records Invoked by: `m-x Select Tag Table`.

**Find Orphan Records**

Finds all records that either have no links to them or no contents regardless of whether they are linked. It searches all documentation systems you have loaded in your world.

## 44.5. The Journal Directories

When your system is compiled, the compiler creates a *Journal Directory* for the bookkeeping it does about the system. The Journal Directory is usually named `patch`; and is located on your system directory. On this directory is a file called `your-system.system-dir` and a subdirectory for each version of the system that has been compiled.

For example:

```
SYS:SCHED;DOC;PATCH;*. *.*
 70911 free, 483779/554690 used (87%, 7 partitions) ...
 5 blocks in the files listed
>sys>sched>doc>patch
  sched-doc.system-dir.5    1  2017(8)    ...
  sched-doc.system-dir.6    1  2117(8)    ...
  sched-doc-1.directory.1   1  DIRECTORY ! ...
  sched-doc-2.directory.1   1  DIRECTORY ! ...
  sched-doc-3.directory.1   1  DIRECTORY ! ...
```

The file `sched-doc.system-dir` contains a lisp form for each version of the system, recording the name of the *component-dir* and the status of the system. The *component-dir* is a file that lists all the files that were compiled into the system. It is located on the subdirectory for that version of the system.

The directory `sched-doc-3.directory` looks like this:

```
Wombat:>sys>sched>doc>patch>sched-doc-3>*. *.*
 70904 free, 483786/554690 used (87%, 7 partitions) ...
 2 blocks in the files listed
  sched-doc-3.component-dir.1  1  2522(8)    ...
  sched-doc-3.patch-dir.1     1  1976(8)    ...
```

This system has not been patched yet, so there are no patch files on the directory. When the system is patched, the patch files (three for each patch, `.lisp`, `.bin`, and `.sab`) go here.

The files on these directories are written automatically. In general it is not a good idea to attempt to edit any of these files by hand. However, you might want to look at the `component-dir` file to check on just exactly which files make up a given system version. This can be useful if a file gets accidentally deleted and you need the version number so it can be reloaded from tape. (Or perhaps, if a file is irretrievably lost, a little sneaky editing of a version number in the `component-dir` to point to a different `.sab` file can salvage things — this is not recommended, but is occasionally necessary).

## 44.6. Distributing Documentation Systems

When you distribute a documentation system with your application software, there are two documentation specific issues that you should know about.

- If you are distributing documentation to a site that is running Genera 7.2, you must distribute NSage patches.
- If you define any customized formatting environments or book designs, you must distribute them or your customers will not be able to read your documentation.

### 44.6.1. Distributing NSage Patches

In addition to your own software (application program and documentation system), if you are distributing documentation to a site running Genera 7.2, you must also distribute the binaries for the additional NSage patches that came on your Symbolics Concordia tape and the NSage patch on the Symbolics Concordia 1.0 ECO #1 tape distributed with this document. These patches are classified as part of development Genera and can, therefore, be distributed under the new Patch Exchange Policy announced to SLUG January 25, 1989 via mail (a copy of which is attached). Please note that distributing these NSage patches is not necessary for sites running only Genera 7.3 Ivory or later releases.

Restore the contents of the Symbolics Concordia 1.0 ECO #1 tape with Restore Distribution. Then load patches for NSage.

To distribute your software, you should use the Distribute Systems Frame and add NSage to the systems you distribute. See the section "Distribute Systems Frame".

Using the Distribute Systems menu frame, with initial default parameters, the NSage system spec should include:

```
NSage, version 27
  Distribute-sources      No
  Distribute-binaries    No
  Included-files-checkpoint  7.2
```

Be sure to mention in your installation instructions that after restoring the distribution tape, the user must load NSage patches.

### 44.6.2. Distributing Book Designs

If you define a new document type or book design that you want to have available online, or if you define your own special-purpose formatting commands or environments that you use in your documentation system, you must be sure to include these in your documentation system and distribute them with your documentation system.

**Note:** The Letter, Article and Memo book designs are defined in Symbolics Concordia, not NSage. They exist for the convenience of users of Symbolics Concordia who want to use Concordia for all their text handling work. If you use these document types for your online documentation, you must be sure to include the definitions for them and their commands as a module in your documentation system and be sure they are included in your distribution.

### 44.6.3. Quality Assurance for Documentation Systems

We recommend that before distributing a documentation system, you do installation and use testing.

Before sending out a documentation system, you should configure a machine at your site with the software that you expect your customers to be using (without Symbolics Concordia) and load your documentation system and the NSage patches. Then use Document Examiner to look at your documentation.

Check that your documents appear in the candidates pane of Document Examiner. If they do not, check your **sage::register-book** forms against the top-level records to make sure the titles are correct.

Use Show Documentation on records that make use of any features such as a document type or screen book design you have designed yourself. Browse through your documentation, displaying a variety of records to make sure that everything you expect to have included is there.

### 44.6.4. Replacing Symbolics Documentation with Your Own Documentation System

When you package an application, you might want to prevent your users from seeing any documentation system except for the one that you provide. You can remove the Symbolics Documentation System from a distribution world you are building for your customers by using **sage::clear-record-registries**. It removes all the index information currently loaded in your world. Then you can load your own documentation system, which is the only information available in Document Examiner.

#### **sage::clear-record-registries**

*Function*

Zeros out all the unique-id registries, the keyword token indices, and the **\*topic-aarray\***. Returns the message "Your NSage doc system has been cleared". This removes all index information and record information currently loaded in your world.

#### 44.6.5. Distributing Documentation for Macintosh Document Examiner

Overview	<p>You can deliver any documentation created with Symbolics Concordia in Document Examiner on the Macintosh. You need only turn it into a Document Set.</p> <p>You first determine the topics that you want to include. After you determine the topics, you create a Macintosh Document Examiner Document Set.</p> <p>You can designate an existing record as a Document Set, for example, a chapter, a section, or an entire book. Additionally, you can designate a Document Set by creating a new record and including all records pertaining to your topic. After you specify a record to serve as the top level record of the Document Set, you use the Write Producer Files command from a Lisp listener to make the Document Set available on the Macintosh desktop.</p>
DocSet Size	<p>Unlike traditional Macintosh documents, you can create a Macintosh Document Set as large as you want. For example, you could create a Document Set including all 10,000 pages of the Encyclopedia Britannica. The Write Producer Files command automatically places all 10,000 pages of this book into one document on the Macintosh desktop as in Figure 72.</p>
Requirements	<p>You can create a Macintosh Document Examiner Document Set and read documentation online using a MacIvory with these features:</p> <ul style="list-style-type: none"><li>• Eight Megabytes of Nubus memory.</li><li>• 320 Megabytes of disk space.</li><li>• The Concordia hypertext documentation system.</li></ul>
Procedure	<p>Follow these steps for creating a Document Set:</p> <ol style="list-style-type: none"><li>1. Make sure you are using a MacIvory running Concordia.</li><li>2. Restore the Producer system to your SYS host by inserting the Producer disk and specifying <code>RESTORE DISTRIBUTION :USE DISK FLOPPY</code>. At the pathname prompt, specify <code>HOST:PRODUCER:DISTRIBUTION.VOLUME-1</code>.</li><li>3. From a Lisp listener, load the Producer system using the Load System command.</li><li>4. Use the Write Producer Files command from a Lisp listener for creating a Macintosh Document set. Specify the</li></ol>

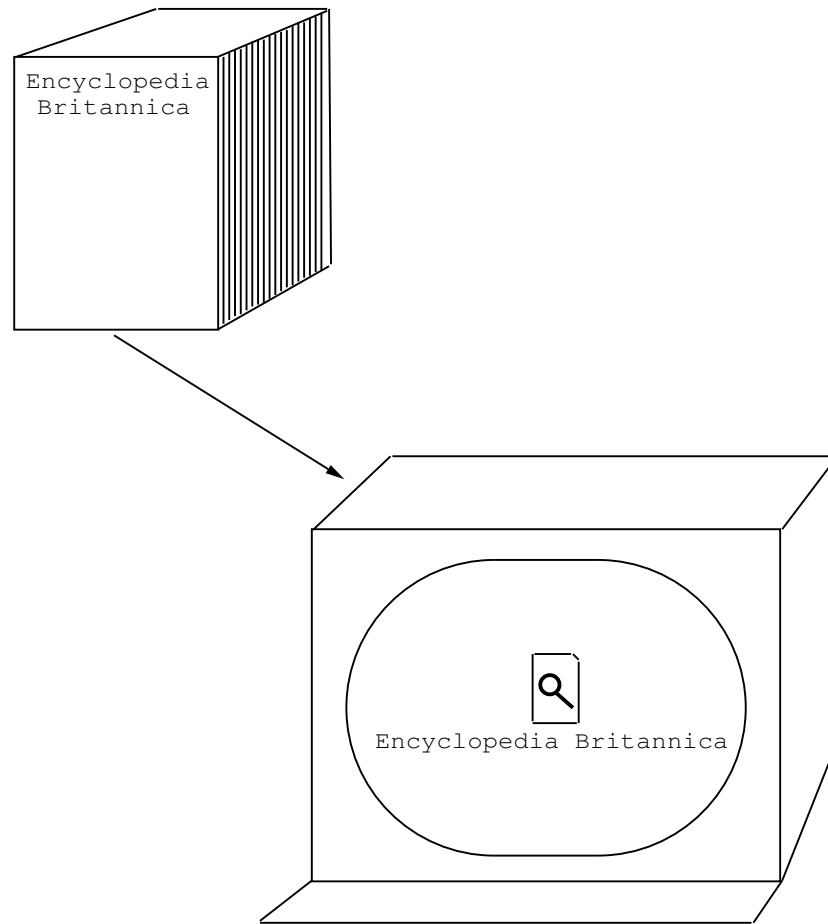


Figure 72. Document Sets.

Document Set name, the top level record of the Document Set, and the Macintosh directory pathname in which the Document Set will reside. For example, you can create a Document Set named **Encyclopedia Britannica** and place in the `host:disk:DEX-Help` directory.

```
Command: Write Producer Files (DocSet Name)
         "Encyclopedia Britannica"
         (a documentation topic) Encyclopedia Britannica
         (a directory pathname) Host:Disk:DEX-Help:
```

You have to perform the first three steps only when creating your first Document Set. When creating subsequent Document Sets, perform step 4 only.



Note If many users view the same Document Sets, you can use the Appleshare server for storing Document Sets.

## Write Producer Files

Write Producer Files *DocSet-Name Documentation-Topic Directory-Pathname keywords*

*DocSet-Name* The name of your Macintosh Document Set. The name that you specify identifies the Document Set that will contain your documentation on the Macintosh. For example, you can create a Document Set named Encyclopedia Britannica. When you complete the Write Producer Files command, a Document Set icon named Encyclopedia Britannica appears on the Macintosh desktop. **Note:** Document Examiner only accepts Document Set names under 32 characters long.

*Documentation-Topic*

A topic including all records that you want to read online on the Macintosh. For example, specifying Encyclopedia Britannica automatically creates a Document Set containing all records included by the topic Encyclopedia Britannica.

*Directory-Pathname* The Macintosh folder in which your Document Set will reside. For example, you can place the Encyclopedia Britannica Document Set in a folder named DEX-Help by specifying *Host:Disk:DEX-Help*: where *Disk* corresponds to the name of your hard disk, and *DEX-Help* corresponds to the name of the folder in which you place your document set.

*keywords* :Format, :Preload-Documentation, :Documentation-Version

:Format {Macintosh, Lisp-Machine} The format of your document set. The default is Macintosh.

:Preload-Documentation {Yes, No} Preloading documentation enables you to load all records necessary for creating a Document Set at one time. For example, if your Document Set contains two hundred records, using this option enables your machine to read all records for the Document Set during one network connection. The default is Yes.

:Documentation-Version {*number*} The version number of the Document Set you are formatting.

Specifying *HOST:DISK:DEX-Help*: as the directory pathname for the Write Producer Files command enables you to place your docu-

mentation into a folder named DEX-Help as in figure 73.

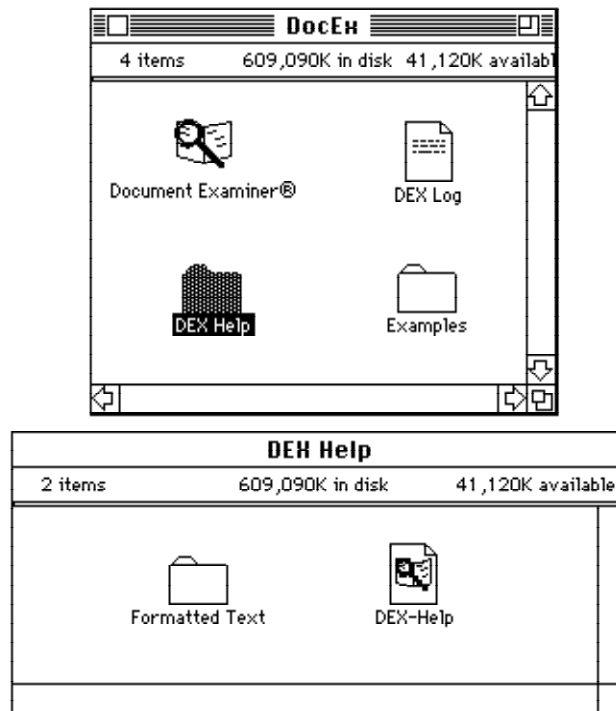


Figure 73. The Macintosh desktop.

#### Considerations

You can store documentation either on your local Macintosh or on a remote Macintosh server. Store your documentation on the machine enabling you to read the documentation most efficiently.

## **PART VIII.**

# **APPLICATION PROGRAMMERS' INTERFACE GUIDE TO SYMBOLICS CONCORDIA**

Symbolics Concordia is built around the Genera application development substrate. Thus all the features of Dynamic Windows, the presentation substrate, **dw:define-program-framework**, and the generic graphics substrate can be used from Symbolics Concordia or to connect to Symbolics Concordia.

Some basic areas where application programs want to hook into the Sage and Symbolics Concordia internals include:

- Documentation lookup and display
- Using commands in documentation records to produce output from a program
- Creating new documentation records
- Displaying graphics drawn by a program
- Creating mouse sensitivity in your output
- Displaying formatted text on a specific device

## 45. Displaying Documentation Under Application Program Control

Your program can display documentation in several ways:

- By using the Show Documentation command.
- By making use of the Extended Help facility which allows you to put documentation on a mouse click, as is done in the Graphic Editor, where clicking middle on a menu item displays the documentation for that item.
- By producing formatted output of generated or other text on the fly.

The simplest way to display documentation is to import the Show Documentation command and any others (Show Overview and Show Candidates, for example) into the command table (comtab) of your program. Then your users can use these commands. To do this you use the function **cp:install-command**. For example:

```
(cp:install-command 'my-program-comtab 'com-show-documentation)
```

A number of Sage internal functions display documentation on the current screen. You can use these internals if your application program knows the documentation topics that it wants to display. For simple interfaces, you can use the interface provided by **cp:execute-command** instead:

```
(cp:execute-command "Show Documentation" "graphics:draw-arrow")
```

**sage:find-record-group-for-topic-and-type** *topic type* &optional *Function*  
*unique-id create-p unique-index supersede-topic-if-*  
*different install-name-p*

Example:

```
(sage::find-record-group-for-topic-and-type 'setq 'sage::function)
(sage::find-record-group-for-topic-and-type "networks" 'sage::section)
```

<i>topic</i>	Either a defined symbol or a string identifying a documentation topic.
<i>type</i>	The set of currently defined types is held in the variable <b>sage:*record-types*</b> . See the examples. The most common type is <b>sage::section</b> ; the rest of the types are either synonyms for section or else types for documenting specific kinds of programming language constructs.
<i>unique-id</i>	The unique identifier given to a record when it is created.
<i>create-p</i>	
<i>unique-index</i>	
<i>supersede-topic-if-different</i>	

*install-name-p*

**sage::lookup-manual-internal** *record-group* &optional (*device-type* *screen*) *Function*

```
(sage::lookup-manual-internal
 (sage::find-record-group-for-topic-and-type
  'setq 'sage::function))
```

**sage:with-sage-device** (*device* &optional *type* &rest *options*) *Function*  
&body *body*

## 46. Finding Topics Under Application Program Control

**sage:candidates-satisfying-condition** *match-list matching-function match-basis condition* *Function*

**sage::doc-match-internal** *match-string matching-function match-basis request-type-string condition &key (:device :screen)* *Function*





## 47. Displaying Overviews Under Application Program Control

**ddex::display-overview-graph** *original-record-group* &optional *Function*  
(*window* \***standard-output**\*)



## 48. Creating Records Under Application Program Control

The first issue you should consider before deciding on how your program should create records is: do you really need to create records or can the information you are collecting from your program be adequately dealt with by **sage:sage-formatting**? To answer this you need to know the expected use of the documentation produced by your program.

- Do you need to store the documentation you have produced, or is its lifetime only for the current program run?
- If the purpose of the program is to convert some information to Symbolics Concordia format for writers to edit using the Symbolics Concordia Editor, you need to create records.
- If your User Interface design requires user look up with Document Examiner, including Show Candidates and Show Overview, you need to create records.

Once you determine that you need to create records, you should determine where these records are going to be created.

- Are they just going to be used in the running world for Document Examiner perusal, or are they going to be stored permanently in files? If they are going to be closely tied to this program run, and recreated for each run, but still need to be examined with the Document Examiner.
- Should they go into buffers for editing, with a writer taking care of saving the files?
- Are they going to be integrated into an existing documentation system (or become a new documentation system)? If so, you need to be sure that you are creating .sab files and they are being properly added to a system declaration, the documentation system is compiled in a timely fashion, and all the other issues of documentation system maintenance. See the section "Guide to Documentation System Maintenance", page 431.

**sage::create-record** &key :topic :type :record-group (:version-number 0)

*Function*

**sage::create-record-internal** *topic-string type buffer &key :exist-* *Function*

*ing-record-ok*

Used by all of the Symbolics Concordia editor commands that create records.

## 49. Displaying Graphics Under Application Program Control

**graphic-editor::\*all-loaded-drawings\*** *Variable*

The graphic editor maintains a list of the drawings that it currently knows about. Each element in the list is an instance of **graphic-editor:drawing**.

Drawings are pushed onto the list as they are created or read in.

**graphic-editor:drawing** *Flavor*

A graphic editor drawing is an instance of **graphic-editor:drawing**. You can display a drawing on any window.

```
(graphics:with-room-for-graphics ()
  (graphic-editor::draw-drawing
    (first graphic-editor::*all-loaded-drawings*
      :stream *standard-output* :if-too-large :clip))
```

This flavor has a number of useful to know about instance variables, which you set with **setf** and interrogate with generic functions.

<b>name</b>	The string used in various commands to identify the drawing.
<b>file</b>	The file in which the drawing has been saved.
<b>modified-p</b>	Indicates whether the drawing has changed since being read in.
<b>entities</b>	The objects that constitute the drawing. This is an array.

**graphic-editor:draw-drawing** *drawing* &key (:stream \*standard-output\*) (:if-too-large :clip) (:presentations t) (:unit :pixel) :float-origin *Function*

Displays an instance of **graphic-editor:drawing** on the specified stream.

This function can show a drawing on a *window*:

```
(graphics:with-room-for-graphics ()
  (graphic-editor::draw-drawing
    (first graphic-editor::*all-loaded-drawings*
      :stream *standard-output* :if-too-large :clip))
```

<i>drawing</i>	An instance of <b>graphic-editor:drawing</b> .
<b>:stream</b>	The stream on which to draw the drawing.
<b>:if-too-large</b>	Specifies what to do when the picture is too large for the paper. The set of possibilities is the same as in the graphic editor <b>Hardcopy</b> command: <b>:clip</b> , <b>:scale</b> , or <b>:multiple-pages</b> .



## 50. Creating Mouse Sensitivity in Your Output

Crossreference links, precis links, and Lisp objects are all mouse-sensitive in your output. In addition, you can make any text mouse-sensitive by using the Invisible appearance of a crossreference (see the section "Crossreference Link", page 89).

You can also use the "Presentation Environment" to introduce arbitrary presentation types, allowing you to create mouse-sensitive items that perform any tasks you can program.

For example, if you were to display this section in a Lisp Listener using the Show Documentation command, the small picture of *Symbolics Concordia* below, when clicked on, would cause *Symbolics Concordia* to be selected.

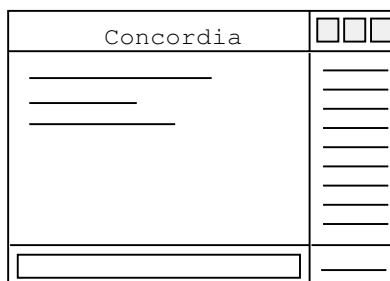


Figure 74. Click on me to select *Symbolics Concordia*.

This is because the picture of the Symbolics Concordia frame is surrounded by a Transparent environment that has the following additional attributes:

```
PresentationObject: (CP:BUILD-COMMAND 'SI:COM-SELECT-ACTIVITY "Symbolics Concordia")
PresentationType: 'CP:COMMAND
```

The PresentationObject associates the picture with the Command Processor command Select Activity, and the picture is now a *presentation* of type **cp:command**. (See the section "The Presentation Type System: an Overview" in *Programming the User Interface*.)

Note, however, that clicking on this picture does not select Symbolics Concordia if you do it in "Document Examiner", because the "Select Activity" command is in the Command Processor's command table but not in "Document Examiner"'s.

To install the "Select Activity" command in "Document Examiner", you must add it to Document Examiner's command table. We can do that here using an ActiveExample:

```
;; The way to determine what symbol names the function
;; that implements a particular command is to do (cp:read-command)
;; and type the command name. You'll get the symbol
;; back as a value.

NIL
(cp:command-table-install-command "Doc-Ex" 'si:com-select-activity)
```

You can click on this example to run the code. First, you can try looking up this documentation section in "Document Examiner" to verify that the picture above is not mouse-sensitive. Then run this example and click on the picture again to verify that it has become mouse-sensitive.

The two environment attributes, **PresentationType** and **PresentationObject** can be added to any environment (or a Transparent environment if you do not want any particular environment). Each of these should be set to a Lisp form that, when evaluated, produces a presentation type or a piece of data, respectively. If you know the presentation type to be a constant, you should quote it with `''`. If you do not specify a **PresentationObject**, the internal Sage datastructure becomes the object.

See the section "Dictionary of Predefined Presentation Types" in *User Interface Dictionary* to use presentation types that are already defined in Genera. If you want a new kind of presentation, see the section "Defining Your Own Presentation Types" in *Programming the User Interface*.

You can create mouse sensitivity for parts of pictures, also. See the section "Making Pictures Using Lisp", page 184.



## 51. Displaying Formatted Text Under Application Program Control

### 51.1. Formatting From a Source with Embedded Formatting Commands

The formatter used by commands like `Format File` is available under program control as well. To use it, you need only a stream in the @-sign markup language described in "Formatting Text in Zmacs". You can yank and run the following example to see how it works:

```
(with-input-from-string (string "This is an example containing
embedded commands for formatting effects like @b(bold)
and @i(italic). More interesting formatting is also possible:
@begin(example)This is a line in an example@end(example)")
(zwei:format-stream (si:make-input-stream-from-tributaries
                    :characters t
                    :string "@begin(text)"
                    :stream string
                    :string "@end(text)")
                    :screen string "An example"))
```



## 52. Application Programmers' Dictionary

**graphic-editor::\*all-loaded-drawings\*** *Variable*

The graphic editor maintains a list of the drawings that it currently knows about. Each element in the list is an instance of **graphic-editor:drawing**.

Drawings are pushed onto the list as they are created or read in.

**graphic-editor:draw-drawing** *Function*  
*drawing* &key (:stream \*standard-output\*) (:if-too-large :clip) (:presentations t)  
 (:unit :pixel) :float-origin

Displays an instance of **graphic-editor:drawing** on the specified stream.

This function can show a drawing on a *window*:

```
(graphics:with-room-for-graphics ()
 (graphic-editor::draw-drawing
  (first graphic-editor::*all-loaded-drawings*)
  :stream *standard-output* :if-too-large :clip))
```

*drawing* An instance of **graphic-editor:drawing**.

**:stream** The stream on which to draw the drawing.

**:if-too-large** Specifies what to do when the picture is too large for the paper. The set of possibilities is the same as in the graphic editor Hardcopy command: **:clip**, **:scale**, or **:multiple-pages**.

**graphic-editor:drawing** *Flavor*

A graphic editor drawing is an instance of **graphic-editor:drawing**. You can display a drawing on any window.

```
(graphics:with-room-for-graphics ()
 (graphic-editor::draw-drawing
  (first graphic-editor::*all-loaded-drawings*)
  :stream *standard-output* :if-too-large :clip))
```

This flavor has a number of useful to know about instance variables, which you set with **setf** and interrogate with generic functions.

**name** The string used in various commands to identify the drawing.

**file** The file in which the drawing has been saved.

**modified-p** Indicates whether the drawing has changed since being read in.

**entities** The objects that constitute the drawing. This is an array.

**sage:candidates-satisfying-condition** *match-list matching-function match-basis condition* *Function*

**sage::create-record** &key :topic :type :record-group (:version-number 0) *Function*

**sage::create-record-internal** *topic-string type buffer &key :existing-record-ok* *Function*

Used by all of the Symbolics Concordia editor commands that create records.

**sage::create-record-type** *type lisp-type print-name external-symbols special-fields* *Function*

Example:

```
(create-record-type :sage-function-record 'defun "Function"
  '(function macro |SPECIAL OPERATOR| |SPECIAL FORM|
    |GENERIC FUNCTION|)
  ("Arglist" "Operation"))

(create-record-type :sage-message-record t "Message" '(message)
  ("Arglist" "Operation"))

;;; Record types for microcode objects. Not "really" Lisp.
(create-record-type :sage-micro-record t "Micro" '(micro)
  ("Arglist" "Operation"))

(create-record-type :sage-section-record nil "Section"
  '(section script chapter subsection
    subsection appendix subheading
    dictionary picture)
  ("Contents"))
```

**sage::define-system-variable** *name accessor* *Function*

Defines new sage system variables. For a list of predefined sage system variables, see the section "Sage System Variables", page 193 .

You can use sage system variables to control document formatting options ( see the section "Setting Document Formatting Options in Symbolics Concordia", page 190 ).

```
(define-system-variable "machine type" #'machine-type)
(define-system-variable "system type"
  #'(lambda () (format nil "~s"
    (send neti:*local-host* :system-type))))
```

*name* A string that names the sage system variable.

*accessor* A function with no arguments that returns a string that is the value of the system variable.

<b>ddex::display-overview-graph</b>	<i>original-record-group</i> &optional ( <i>window</i> * <b>standard-output</b> *)	Function
<b>sage::doc-match-internal</b>	<i>match-string matching-function match-basis request-type-string condition</i> &key (:device :screen)	Function
<b>sage::doc-record-for-record-group</b>	<i>record-group</i> &optional ( <i>monitor t</i> )	Function
<b>ddex::dynamic-dex-doc-match-internal</b>	<i>match-string matching-function match-basis reason-string condition</i> of <b>ddex::doc-ex</b>	Method
<b>sage::find-record-for-lookup</b>	<i>record-group</i> &key :error-p :loading-contents	Function
<b>sage:find-record-group-for-topic-and-type</b>	<i>topic type</i> &optional <i>unique-id create-p unique-index supersede-topic-if-different install-name-p</i>	Function

## Example:

```
(sage::find-record-group-for-topic-and-type 'setq 'sage::function)
(sage::find-record-group-for-topic-and-type "networks" 'sage::section)
```

<i>topic</i>	Either a defined symbol or a string identifying a documentation topic.
<i>type</i>	The set of currently defined types is held in the variable <b>sage:*record-types*</b> . See the examples. The most common type is <b>sage::section</b> ; the rest of the types are either synonyms for section or else types for documenting specific kinds of programming language constructs.
<i>unique-id</i>	The unique identifier given to a record when it is created.
<i>create-p</i>	
<i>unique-index</i>	
<i>supersede-topic-if-different</i>	
<i>install-name-p</i>	

**sage:in-environment** (*stream envr-name* &rest *envr-mods*) &body *body* Macro

Does *body* with a Sage environment in effect, as named by *envr-name* and adjusted by *envr-mods*, which are attribute/value pairs. The environment name and the attribute/value pairs are not evaluated.

**sage:in-environment1** (*stream envr-name&mods-list*) &body *body* *Macro*  
 Like `in-environment`, except that the environment name and modification list, (**name attribute value...**), is a computed list. You may want to use backquote.

**sage:in-item** (&optional *stream*) &body *body* *Macro*  
 In a description or itemize environment, put each item in (**in-item** () ...). This hides the implementation detail of paragraph breaks.

**sage:in-presentation** (&optional *stream* &key *:type :object :presentation-options*) &body *body* *Macro*  
 Causes the output of *body* to be turned into a presentation with associated object *object* and type *type*.

**sage::lookup-manual-internal** *record-group* &optional (*device-type :screen*) *Function*  

```
(sage::lookup-manual-internal
 (sage::find-record-group-for-topic-and-type
 'setq 'sage::function))
```

**sage::make-sage-record** *raw-fields* *Function*  
 In spite of its attractive name, this has nothing to do with creating records in Symbolics Concordia. The records that it is prepared to create are ones based on a textual representation with embedded commands.

**sage:making-sage-directives** (&optional *stream*) &body *body* *Macro*  
 Binds *stream* to a special stream, which collects all formatting and output done in *...body...*, making Sage directive structure out of them, which it returns. It must run in a **sage:with-sage-device** dynamic environment.

**sage:record-group** *Presentation Type*  

```
(define-test-command (com-show-doc :menu-accelerator "Show"
                                   :menu-level (:top-level (:mac :lookup))
                                   :keyboard-accelerator #\s-S)
 ((record-group 'sage:record-group :prompt "for topic"))
 (rpc:with-output-to-viewer
 (t :title (present-to-string record-group 'record-group))
 (sage::with-sage-device-for-mac-stream (dev)
 (sage::lookup-manual-internal record-group dev)))
 )
```

**sage::record-group-callers** *record-group* *Function*

**sage:sage-command** *stream symbol* &rest *params* *Function*

For things like `(sage-command stream 'tab-to-tab-stop)` which is certainly more verbose (and differently cryptic) than `@\`.

- sage:sage-contents-list** *stream contents-list* *Function*  
 When you already have a computed Sage contents-list, emit it with  
 (**user::sage-contents-list** *stream contents-list*)
- sage:sage-formatting** (*&optional body-stream &rest sage-device-  
 options*) *&body body* *Macro*  
 Within *body*, all the output to *body-stream*, together with the formatting  
 macros herein, gets collected, passed through the Sage formatter and dis-  
 played. It uses **sage:making-sage-directives**.
- sage:sage-paragraph** *&optional stream* *Function*  
 Separate paragraphs (as opposed to enveloping a paragraph) by emitting  
 something so Sage thinks there is a paragraph break here. Today, that  
 means two #\Return characters. **sage:in-item** calls **sage:sage-paragraph**.
- sage:sage-tab** *&optional stream* *Function*  
 Emits a tab character that gets converted to tab-to-tab-stop.
- sage::set-static-value** *name value* *Function*  
 Sets or clears a sage static variable. Sage static variables are used to con-  
 trol document formatting options. For more information, see the section  
 "Setting Document Formatting Options in Symbolics Concordia", page 190.
- |              |   |
|--------------|---|
| <i>name</i>  | A string that names the static variable.  |
| <i>value</i> | A string that is the value of the static variable. A <i>value</i><br>of <b>nil</b> clears the variable. |
- sage:with-sage-device** (*device &optional type &rest options*) *Function*  
*&body body*





## 53. Replacing Symbolics Documentation with Your Own Documentation System

When you package an application, you might want to prevent your users from seeing any documentation system except for the one that you provide. You can remove the Symbolics Documentation System from a distribution world you are building for your customers by using **sage::clear-record-registries**. It removes all the index information currently loaded in your world. Then you can load your own documentation system, which is the only information available in Document Examiner.

### **sage::clear-record-registries**

*Function*

Zeros out all the unique-id registries, the keyword token indices, and the **\*topic-array\***. Returns the message "Your NSage doc system has been cleared". This removes all index information and record information currently loaded in your world.



## Index

.sab files and buffers, 65  
@#, 407  
Above Attribute, 137, 170  
Abovebelowstyle Environment, 418  
Adding Formatting Markup to Your Records, 38  
Add Record Field, 76  
Advantages of Symbolics Concordia's Linking  
    Strategy, 100  
:**advertised-in** option for **defsystem**, 432  
An Example of a Modification to an Environment  
    with Dependencies, 378  
An Example of a Modification to a Top-level  
    Specification in a Document Device  
    Type, 380  
An Example of a Simple Modification, 374  
An Example of Writing for Modularity, 211  
AppendixSection Command, 429  
Application Programmers' Dictionary, 473  
Application Programmers' Interface Guide to  
    Symbolics Concordia, 458  
Arglist field, 73  
argument list, 73  
A System Declaration for a Large Document Set,  
    436  
Attributes of an environment, 113  
Below Attribute, 137, 170  
Bitmap Editor Basic Concepts, 319  
bitmap record markers, 183  
Blanklines Attribute, 137, 170  
Bodystyle Environment, 418  
Bogus Links, 448  
book, 357  
book design, 357  
Book Design Browser, 369  
book design element, 358  
Book Design Elements, 366  
Book Design Functions and Variables, 421  
Book Design Functions Dictionary, 418  
Book Design Source Files, 362  
book design sources, 359  
Book Design Vocabulary, 357

Bottommargin Attribute, 170  
Boxbm Attribute, 170  
BoxFlushRight Attribute, 170  
Boxlm Attribute, 170  
Boxrm Attribute, 170  
Boxtm Attribute, 170  
Boxtype Attribute, 171  
Break Attribute, 171  
**:bug-reports** option for **defsystem**, 434  
Bugs in documentation, 281  
Building a Document: Linking Records, 85  
bullet, 120  
C-U S-M, 119  
C-U S-P, 41  
Can Decompressing the Documentation Database  
    Be Skipped?, 18  
Capitalized Attribute, 171  
Caption Command, 138  
Capturing Screen Images, 322  
Case Command, 11  
Cat and Mouse, 179  
Centered Attribute, 171  
Center Environment, 129  
Center View Bitmap Editor Command, 322  
Change Entity, 11  
Change Layout Page Previewer command, 206  
Change Record Type command, 71  
Change Record View, 83  
Changing a Markup Command, 119  
Changing an environment and/or its attributes, 113  
Changing Graphic Editor Defaults, 240  
Changing the Capitalization and Character Style of  
    a Record Topic, 80  
Changing the Organization of Your Document, 110  
Changing the Title of a Record , 37  
Changing the View of Drawings, 265  
Changing the Way a Record Topic Displays, 80  
Chapter Command, 429  
Characteristics of Mark-up Environments, 113  
Character Styles in Symbolics Concordia, 174  
Checking Your Documentation System, 448  
Checklist Environment, 125  
Clean Directory (n-X), 446  
Clean File, 448  
Clean File (n-X), 446  
Cleaning up, 446  
Clear All Bitmap Editor Command, 320

Clear Display, 412  
Clear Gray Bitmap Editor Command, 320  
Clear Record Name Prefix Command, 347  
Clear Record Name Prefix command, 71  
Clear Record Name Prefix Command, 8, 11  
Clear Record Name Suffix Command, 347  
Clear Record Name Suffix command, 71  
Clear Record Name Suffix Command, 8, 11  
Clear Register Bitmap Editor Command, 321  
Clear Sage Variable Command, 11, 209  
Collected Record Names Pane, 39  
collectors, 359  
Collect Record Name command, 81  
Columnmargin Attribute, 171  
Columns Attribute, 171  
Columnwidth Attribute, 171  
Combining Entities to Produce Complex Shapes,  
275  
command, 359  
commands for creating records, 65  
Compiling, 431  
Compiling a Documentation System, 438  
Compiling a Documentation System for Multiple  
Machine Types, 440  
Concordia Example: A Book with Multiple  
Versions, 193  
Concordia Markup Language, 119  
Contents field, 73  
Contents Link, 87  
Contents link, 86  
Contentsstyle Environment, 418  
Continue Attribute, 171  
Controlling Mouse-Sensitivity in Graphic Editor  
Drawings, 245  
Controlling the Size and Alignment of Entities, 283  
Controlling Your Document's Appearance, 113  
Convert Flat Text to Record Command, 8, 11  
Converting Documents From Concordia to Scribe,  
353  
Converting Existing Documentation to Concordia,  
342  
Converting Graphics to Symbolics Concordia, 349  
Converting Scribe Documents to Symbolics  
Concordia, 345  
Converting Text Files to Symbolics Concordia, 347  
Convert Topic to Scribe, 353  
Copy Drawing command, 266

- Copy Drawing to Image Graphic Editor Command,  
11
- Copying Drawings, 266
- counter, 359
- Counter templates, 407
- Count Records in Buffer Command, 11
- CRBreak Attribute, 171
- Create Link and Record command, 71, 105
- Create Picture, 179
- Create Record command, 71, 105
- Creating Active Examples, 183
- Creating a Diagram, 43
- Creating a File for Your New Record, 25
- Creating a Letter Document Type, 393
- Creating an Article Document Type, 405
- Creating and Filling in a Documentation Record, 71
- Creating an Environment, 116
- Creating a New Document Type, 390
- Creating a New Stipple Pattern, 335
- Creating an Index in a Symbolics Concordia  
Document, 187
- Creating a Screen Snapshot, 48
- Creating Documents with Symbolics Concordia, 65
- Creating Drawings with the Graphic Editor, 259
- Creating Mouse Sensitivity in Your Output, 469
- Creating new markup types, 393
- Creating Non-Standard Shapes, 273
- Creating Records for the Topics in Your Outline, 33
- Creating Records Under Application Program  
Control, 465
- Creating Shapes with the Graphic Editor, 249
- Creating Simple Tables, 135
- Creating the Record, 26
- Creating Title Pages, 388
- Creating Your Own 3Symanual Document Type,  
404
- Creating Your Own Crossreference Appearance,  
399
- Crossreference, 86
- Crossreference Link, 89
- Crossreference link, 86
- Crossreference Page Numbers, 206
- CRSpace Attribute, 171
- Customizing the Editor Pane, 215
- Customizing the Symbolics Concordia Window, 61
- Customizing Your Symbolics Concordia  
Documents, 189
- ddex::display-overview-graph** function, 463, 475

**ddex::dynamic-dex-doc-match-internal** method  
    of **ddex::doc-ex**, 475

Decompression Procedure, 18

**:default-module-type** option for **defsystem**, 433

**:default-package** option for **defsystem**, 433

**:default-pathname** option for **defsystem**, 432

Defining a Book Design System for a Site, 410

Defining Counters for an Article, 405

Defining Headings for an Article, 408

Delete Printer Request Command, 209

Deleting an Old Version of Your Documentation  
    System, 447

Deleting Documentation Records, 83

Deleting Extra Versions of Files, 448

Describe Book Design Element, 360, 412

Describe Document Device Type, 360, 413

Describe Environment, 360, 414

Description Environment, 126

device type, 358

Dictionary of Bitmap Editor Commands, 327

Dictionary of Book Design Browser Commands,  
    412

Dictionary of Graphic Editor Commands, 293

Dictionary of Stipple Editor Commands, 336

Dictionary of Symbolics Concordia Editor  
    Commands, 219

Dictionary of Symbolics Concordia Markup  
    Environments and Commands, 141

Directory-Pathname, 455

Disk Space, 446

Display Environment, 128

Displaying and Reviewing Documentation Online  
    , 197

Displaying Documentation, 198

Displaying Documentation Under Application  
    Program Control, 459

Displaying Formatted Text Under Application  
    Program Control, 471

Displaying Graphics Under Application Program  
    Control, 467

Displaying Overviews Under Application Program  
    Control, 463

Displaying Previously Formatted Pages, 205

Displaying, Reviewing, and Publishing  
    Documentation, 197

Display-name field, 79

**:distribute-binaries** option for **defsystem**, 433

**:distribute-sources** option for **defsystem**, 433

Distributing Book Designs, 452  
Distributing Documentation for Macintosh  
    Document Examiner, 453  
Distributing Documentation Systems, 451  
Distributing NSage Patches, 451  
DocSet-Name, 455  
Documentation Conventions, 1  
documentation database, 65  
Documentation Database Compression and  
    Decompression, 17  
Documentation-Topic, 455  
:Documentation-Version, 455  
document device type, 358  
Document Device Types, 363  
Document Examiner, 65  
Document Examiner customizations, 215  
document type, 358  
Drawing, 319  
Drawing files in a Documentation System, 431  
Drawing Irregular Shapes, 273  
Duplicate definitions, 448  
DynamicText Command, 11  
Edit Book Design Element, 416  
Edit Defaults, 11  
Editing Bitmap Images, 326  
Editing the Name of a Symbolics Concordia  
    Record, 79  
Editor Commands pane customizations, 215  
Edit Record in the Page Previewer, 206  
Edit Rulers, 283  
Entering Bitmap Editor Commands, 318  
Entering the Diagram Into a Record, 46  
Entering the Screen Snapshot Into a Record, 50  
Enumerate Environment, 123  
environment, 359  
Environment Attributes, 117  
Environments and How to Use Them, 113  
Examining and Changing the Organization of a  
    Document, 109  
Examining the Organization of Your Document,  
    109  
Example Environment, 128  
Example environment, example, 128  
example of a filled-in record, 73  
Example of Capturing a Specified Rectangle of a  
    Screen, 324  
Example of Using the Graphic Editor, 286  
example record markers, 183



Examples of Document Device Type Modifications,  
374

FaceCode Attribute, 171

Fill Attribute, 171

Filled environments, 113

Filling a Shape Entity with a Pattern, 285

Filling in the Contents of Your Records, 35

Filling in the Keywords Field, 31

Filling in the Oneliner and Keywords Fields, 29

Filling in the Oneliner Field, 29

Finding Topics Under Application Program Control,  
461

Fixed Attribute, 171

Fixing Formatting Errors, 205

Float Attribute, 171

FloatPage Attribute, 171

Flushleft Attribute, 171

Flushleft Environment, 129

FlushRight Attribute, 172

Flushright Environment, 129

Fnenv Environment, 418

Font Attribute, 172

:Format, 455

Format Environment, 127

Format Pages Page Previewer command, 206

Format Pages Page Previewer Command, 11

Formatting a document, 201

Formatting From a Source with Embedded  
Formatting Commands, 471

Formatting of crossreferences, 89

Formatting of precis links., 92

Formatting Page Headings, 403

Formatting Pages, 202

Formatting Tables of Contents and Lists of Figures,  
401

Free Attribute, 172

FTG Environment, 420

Getting Help in Symbolics Concordia, 70

Getting Help in the Book Design Browser, 371

Getting Started in Symbolics Concordia, 68

Getting Started with the Graphic Editor, 237

Give Image To Graphic Editor, 317

**graphic-editor::\*all-loaded-drawings\*** variable,  
467, 473

**graphic-editor:draw-drawing** function, 467, 473

**graphic-editor:drawing** flavor, 467, 473

Grid Size, 283

Group Attribute, 137, 172

Grouping Entities, 279

Guide to Documentation System Maintenance, 431

Guide to Symbolics Concordia Book Design, 357

Handling Tables of Contents and Frontmatter, 387

Hardcopying a formatted document, 205

Hardcopy Pages Page Previewer command, 206

Hd0 Environment, 420

Hd1a Environment, 420

Hd1 Environment, 420

Hd2 Environment, 420

Hd3 Environment, 420

Hd4 Environment, 420

Hdg Environment, 420

HELP command, 70

Hierarchy of a Book, 97

How the Book Design Sources Are Organized, 362

How the Formatter Processes a Record, 95

Hyphenbreak Attribute, 172

Include Link, 86

Include link, 86

Including a Graphic Editor Drawing in a Document,  
270

Increment Attribute, 172

Indentation Attribute, 172

Indent Attribute, 137, 172

Indexenv Environment, 419

IndexStyle, 419

Inheritance Dependency, 359

**:initial-status** option for **defsystem**, 434

Inserting a Bitmap Image Into a Graphic Editor  
Drawing, 269, 325

Inserting a Markup Command, 118

Inserting a Tab character, 118

Inserting Illustrations in Your Symbolics Concordia  
Documents, 179

Inserting internal markup, 119

Internal and External Markup, 119

Internally Used Formatting Styles, 418

Internally Used Heading Styles, 420

Internally Used Table of Contents Entry Styles, 421

Introduction, 343, 373

Introduction to Modular Writing, 211

Introduction to Symbolics Concordia Book Design,  
357

Introduction to Symbolics Concordia Markup  
Language, 113

Introduction to the Book Design Browser, 369

Introduction to the Symbolics Concordia Writer's Guide, 59

Invisible-style Crossreferences, 91

Issuing Commands in the Book Design Browser, 372

Itemize Environment, 120

**:journal-directory** option for **defsystem**, 434

Justification Attribute, 172

keywords field, 187

Keywords field, 73

Keywords field and the index, 73

Killing a Markup Command, 119

Killing an Environment and Environment Markup, 117

Killing environment markup, 113

Killing environment markup and contents, 113

Largestyle Environment, 418

Layout of the Book Design Browser Window, 370

LeadingSpaces Attribute, 172

Leftmargin Attribute, 137, 172

Linewidth Attribute, 172

Link Commands, 105

Linking the Records, 39

Link Types, 86

lispm-init, 215

List Book Design Elements Using Environment, 416

List of Symbolics Concordia Attributes, 170

List Sage Variables Command, 11, 209

Loading Concordia on 3600-Family Machines, 16

Locating Records, 81

Longlines Attribute, 137, 172

Looking At the Document, 41

Looking at the Record Template, 26

**:maintaining-sites** option for **defsystem**, 433

Maintaining Your Documentation Sources, 446

MajorPart Command, 429

Make Active Text Command, 11

making lists, 120

Making Lists in Concordia, 119

Making parts of pictures mouse-sensitive, 184

Making Pictures Using Lisp, 184

Making Sure the System Declaration is Correct, 439

Managing Documentation Records, 79

Managing Drawings, 265

Marking up Lisp Language Objects, 175

Markup Commands and How to Use Them, 118

Merge Gray Bitmap Editor Command, 320  
Modifier Key Conventions, 2  
Modifying a Document Device Type, 373  
Modifying a Single Book, 384  
Modifying a Single Instance of an Environment,  
389  
Modifying the Appearance of Your Documentation,  
361  
Mouse Sensitivity in the Book Design Browser, 373  
Move Black Bitmap Editor Command, 320  
Move Gray Bitmap Editor Command, 320  
Moving Among the Symbolics Concordia Editor,  
Page Previewer, Graphic Editor, and  
Book Design Browser, 178  
Moving Around Symbolics Concordia, 177  
Moving Around the Symbolics Concordia Buffer,  
177  
Moving Records, 81  
Moving to Another Page, 204  
m-x Create Example Record Marker, 183  
m-x Test Example, 183  
Naming commands and environments, 393  
NarrowestBlank Attribute, 172  
Need Attribute, 173  
Next Page Page Previewer command, 206  
Notestyle Environment, 418  
Numbered Attribute, 173  
numbered list, 123  
Numberfrom Attribute, 173  
Numbering Templates, 407  
NumberLocation Attribute, 173  
Oneliner field, 73  
Online Reading Path, 102  
Orphan record, 85  
Orphans, 448  
outline form, 123  
:Output Destination, 330, 333, 350, 353  
Overall Book Structure, 99  
Overview, 235  
Overview of Symbolics Concordia 3.1, 7  
Overview of Symbolics Concordia Documentation,  
1  
Overview of the Page Previewer, 201  
Overview of the Screen Hardcopy Menu, 323  
Packages for Documentation Systems, 435  
Pagebreak Attribute, 173  
Page Previewer Commands, 206  
Paragraphbreaks Attribute, 173

Parse and Replace Region command, 113  
**:patchable** option for **defsystem**, 432  
Patching a Documentation System, 442  
PermanentString Command, 11  
perspective, 275  
picture record markers, 183  
Picturescale Attribute, 173  
Planning Your Document, 33  
**Point not in interval being displayed.**, 217  
Positioning Text in Concordia, 129  
Precis Link, 92  
Precis link, 86  
Preface to Symbolics Concordia, 1  
Preface to the Symbolics Concordia Workbook, 20  
:Preload-Documentation, 455  
Presentationobject Attribute, 173  
Presentationtype Attribute, 173  
Presentation Types for Mouse Sensitive Text and Drawings, 247  
Press file, 206  
**:pretty-name** option for **defsystem**, 432  
Prevailing Scale, 180  
Previous Page Page Previewer command, 206  
Printing a document, 41  
Printing Draft Copies of a Document, 199  
Printing Formatted Pages, 205  
Printing the Record, 32  
Problems and Solutions for Modular Writing, 213  
Producing a Document as a Book, 201  
Proofreading Your Work, 198  
Publishing Your Documentation, 199  
Putting Pictures in Text, 179  
Putting Some Text in the Record, 27  
Quality Assurance for Documentation Systems, 452  
Quantize Mouse, 283  
Quick Conversion for Producing Printed Books, 351  
Read File, 268  
Read Image File Command, 349  
Reap Protecting the New Version, 440  
record fields, 73  
Record-name field, 79  
Record Registry, 65  
record template, 73  
Recovering From Errors in Symbolics Concordia, 217  
Ref Command, 139

Referenced Attribute, 173  
 Refit Box, 317  
 Registering Your Documentation System, 435  
 Registers, 266  
 Releasing a System, 445  
 Rename Entity, 275  
 Rename Record command, 79  
 Renaming a Record, 79  
 Replacing Symbolics Documentation with Your  
     Own Documentation System, 452,  
     479  
 Reset View, 265  
 Resolving Crossreferences, 202  
 Resolving page numbers for references, 202  
 Restart Printer Request Command, 209  
 Restore Tags, 202  
 Restore Tags Page Previewer command, 202  
 Retrieve Register Into Black Bitmap Editor  
     Command, 320, 321  
 Retrieve Register Into Gray Bitmap Editor  
     Command, 320, 321  
 Reviewing documentation in Document Examiner,  
     197  
 Reviewing documentation in the Concordia editor,  
     197  
 Reviewing Work in Document Examiner, 197  
 Reviewing Work in the Symbolics Concordia Editor,  
     197  
 Rightmargin Attribute, 137, 173  
 Run-in Markers, 114  
 Running heads, 139, 385  
 s- = keyboard accelerator, 118  
 s- > Command, 130, 134  
 s- > keyboard accelerator, 118  
 s-HELP command, 70  
 s-m keyboard accelerator, 118  
 s-sh-Middle Symbolics Concordia command, 83  
 s-TAB Command, 130, 133  
 s-TAB keyboard accelerator, 118  
 s= Command, 130, 134  
**sage::\*allow-index-commands-from-record-  
     titles\*** variable, 421  
**sage:\*default-device-type\*** variable, 421  
**sage:\*default-document-type\*** variable, 421  
**sage:\*lgp2-default-character-style\*** variable, 428  
**sage:\*record-lookup-mode\*** variable, 215  
**sage:\*print-package\***, 435  
**sage::register-book**, 431, 436

**sage:define-documentation-system**, 431  
**sage:candidates-satisfying-condition** function, 461, 474  
**sage::clear-record-registries** function, 452, 479  
**sage::create-record** function, 465, 474  
**sage::create-record-internal** function, 466, 474  
**sage::create-record-type** function, 474  
**sage:define-book-design** function, 391, 421  
**sage:define-box-type** function, 424  
**sage:define-crossreference-appearance** macro, 400, 424  
**sage::define-device-type** function, 425  
**sage::define-document-type** function, 425  
**sage:define-line-type** function, 425  
**sage:define-sage-attribute** function, 425  
**sage:define-sage-command** function, 425  
**sage::define-sage-counter** function, 426  
**sage::define-sage-font** function, 426  
**sage::define-system-variable** function, 474  
**sage::describe-environment** function, 426  
**sage::doc-match-internal** function, 461, 475  
**sage::doc-record-for-record-group** function, 475  
**sage::find-record-for-lookup** function, 475  
**sage:find-record-group-for-topic-and-type** function, 459, 475  
**sage:graph-book-design-users** function, 426  
**sage:graph-book-design-uses** function, 427  
**sage:idders** macro, 427  
**sage:in-environment** macro, 475  
**sage:in-environment1** macro, 476  
**sage:in-item** macro, 476  
**sage:in-presentation** macro, 476  
**sage:load-doc-database-for-writer** function, 18  
**sage::lookup-manual-internal** function, 460, 476  
**sage::make-sage-record** function, 476  
**sage:making-sage-directives** macro, 428, 476  
**sage:note-book-design-specifics** function, 428  
**sage:record-group** presentation type, 476  
**sage::record-group-callers** function, 476  
**sage::register-book** function, 12, 428  
**sage:sage-command** function, 476  
**sage:sage-contents-list** function, 477  
**sage:sage-formatting** macro, 477  
**sage:sage-paragraph** function, 477  
**sage:sage-tab** function, 477  
**sage::set-static-value** function, 477  
 Sage System Variables, 193  
**sage:with-sage-device** function, 460, 477

Save Tags, 202  
Save Tags Page Previewer command, 202  
Saving Tags, 206  
Scaling, 283  
Scaling all pictures in a document, 180  
Script Attribute, 173  
script record, 97  
Section Command, 429  
Sectioning Commands for Conversion  
    Compatibility, 429  
SectionRef Command, 430  
Selecting Entities, 280  
Selecting the Symbolics Concordia Editor, 23  
Set Drawing File, 268  
Set Page Page Previewer command, 206  
Set Record Name Prefix Command, 347  
Set Record Name Prefix command, 71  
Set Record Name Prefix Command, 8, 12  
Set Record Name Suffix command, 71  
Set Record Name Suffix Command, 8, 12, 347  
Set Sage Variable Command, 12, 209  
Setting and Using Tabs in Concordia, 130  
Setting a System's Status, 445  
Setting Document Formatting Options in Symbolics  
    Concordia, 190  
Setting Off Text in Concordia, 127  
Setting the Buffer Locking Mode in Symbolics  
    Concordia, 216  
Setting the Lookup Mode in the Document  
    Examiner, 215  
Setting up the Hierarchy of a Book, 99  
Setting up Your Documentation System, 431  
Set Widow Action Page Previewer command, 206  
Show Dependencies on Book Design Element,  
    360, 416  
Show Dependencies on Environment, 360, 417  
Show Documentation in the Page Previewer, 206  
Show File Drawings, 268  
Show Formatted Pages Page Previewer command,  
    206  
Show Improperly Referenced Tags, 202  
Show Improperly Referenced Tags command, 202  
Show Overview in Document Examiner, 73  
Show Overview in the Page Previewer, 206  
Show Printer Status Command, 209  
Simpletable, 135  
Simpletablespecs, 135  
Sink Attribute, 173



Size Attribute, 173

Some Commonly Used Environments, 119

Spaces Attribute, 173

Spacing Attribute, 138, 173

Spacing, punctuation, and capitalization in  
    crossreferences, 89

specification, 357

Specifying Page Headings and Footings, 139, 385

Specifying the Size of Pictures, 180

Spread Attribute, 137, 174

Store Black Into Register Bitmap Editor Command,  
    321

Store Gray Into Register Bitmap Editor Command,  
    321

Storing Drawings in Files, 267

Strategies and Hints for Using the Graphic Editor,  
    273

Strategies for Patching Documentation, 443

SubSection Command, 430

SubSubSection Command, 430

Summary of Compiling the Documentation  
    Database, 442

Summary of New Commands for Symbolics  
    Concordia 3.1, 11

Suppress Index, 206

Suppress Table of Contents, 206

Swap Gray Bitmap Editor Command, 321

Swap Region Bitmap Editor Command, 321

Symbolics Concordia 3.1 Release Notes, 6

Symbolics Concordia Administrator's Guide, 356

Symbolics Concordia Customizations for Your Init  
    File, 215

Symbolics Concordia documentation record, 71

Symbolics Concordia Installation Guide, 14

Symbolics Concordia Installation Procedure, 15

Symbolics Concordia Record Fields, 73

Symbolics Concordia Reference Card, 231

Symbolics Concordia Workbook, 20

Symbolics Concordia Workbook: Creating a  
    Record, 23

Symbolics Concordia Workbook: Creating a Short  
    Document, 33

Symbolics Concordia Workbook: Creating  
    Graphics, 43

Symbolics Concordia Workbook: Viewing Your  
    Document in the Page Previewer, 53

Symbolics Concordia Writer's Guide, 58

- System Status for Loading and Other Operations, 446
- System Status in the Herald, 445
- Tabclear command, 130
- TabClear Command, 130
- TabDivide Command, 130
- TabExport Attribute, 174
- TabSet Command, 130, 132
- Tag Command, 138
- Tc0 Environment, 421
- Tc1 Environment, 421
- Tc3 Environment, 421
- Tcc Environment, 421
- Tcx Environment, 421
- Techniques for Using the Stipple Editor, 336
- The Actual Compilation, 439
- The array given to the ARRAY-LEADER instruction .....**, 218
- The Graphic Editor Screen Layout, 237
- The Journal Directories, 450
- The Most Common Environment Attributes, 137
- The Most Common Markup Commands, 138
- The Significance of Links, 95
- The Symbolics Concordia Window, 59
- The System Declaration for a Documentation System, 431
- tick-mark, 120
- Tips and Techniques for Using Pictures in Documents, 182
- Titlestyle Environment, 418
- Topic Crossreferences, 91
- top-level book design element, 358
- top-level specification, 358
- TopMargin Attribute, 174
- Transformations on Entities, 281
- typescript record markers, 183
- Underline Attribute, 174
- Unexpected selections, 280
- Unfilled environments, 113
- UnNumbered Attribute, 174
- Unparse and Replace command, 113
- Update Arglist Field command, 73
- Updating the System Declaration, 435
- Use Attribute, 174
- Using Crossreferences in Modular Writing, 212
- Using Documentation Records, 65
- Using Drawings in Symbolics Concordia Documents, 269

Using Filling to Produce Shapes, 277

Using Links to Create Tables, 92

Using Links to Incorporate Text, 86

Using Links to Refer to Other Records, 89

Using Registers to Copy Images, 321

Using Sage Static Variables to Set Document  
    Formatting Options, 192

Using Sage System Variables to Set Document  
    Formatting Options, 193

Using Screen Images as Illustrations, 322

Using the Bitmap Editor, 317

Using the Bitmap Editor Window, 317

Using the Collected Record Names Pane, 104

Using the Correct Spacing for Crossreference  
    Links, 91

Using the Graphic Editor, 234

Using the Gray and Black Planes to Edit Images,  
    320

Using the Mouse in the Graphic Editor, 238

Using the Page Previewer, 201

Using the Stipple Editor, 335

Value Command, 12

values record markers, 183

Verbatim Environment, 127

Viewing Bitmap Images, 321

Viewing Records, 83

What is Book Design?, 357

What is Documentation Database Compression?,  
    17

When Markup Damages the Undo History, 218

When Symbolics Concordia Misplaces the Cursor,  
    217

When You Cannot Patch, 444

Why Decompress the Documentation Database?,  
    17

Why Patching Documentation is Important, 443

Widestblank Attribute, 174

**Window start-bp not in interval being  
    displayed., 217**

Within Attribute, 174

Working on a Document Outside Its Documentation  
    System, 444

Workstyle Issues in Writing with Symbolics  
    Concordia, 211

Write File, 267

Write Producer Files, 455

Zoom, 265

Zoom/Expand Bitmap Editor Command, 322

Zoom by Factor Bitmap Editor Command, 322

**zwei:\*concordia-buffer-disposition\*** variable,  
216