

PLT Framework: GUI Application Framework

Robert Bruce Findler (robby@plt-scheme.org)
Matthew Flatt (mflatt@plt-scheme.org)

301
Released December 2006

Copyright notice

Copyright ©1996-2005 PLT

Permission to make digital/hard copies and/or distribute this documentation for any purpose is hereby granted without fee, provided that the above copyright notice, author, and this permission notice appear in all copies of this documentation.

Contents

1 This Manual	1
1.1 Thanks	1
2 Overview	2
3 Preliminaries	3
3.1 Libraries	3
3.2 Mixins	3
3.3 GUI Test Suite Utilities	4
3.3.1 Actions and completeness	4
3.3.2 Errors	5
3.3.3 Technical Issues	5
4 Application	6
5 Autosave	7
5.1 <code>autosave:autosavable<%></code>	7
6 Canvas	8
6.1 <code>canvas:basic<%></code>	9
6.2 <code>canvas:basic-mixin</code>	9
6.3 <code>canvas:color<%></code>	10
6.4 <code>canvas:color-mixin</code>	10
6.5 <code>canvas:delegate<%></code>	10
6.6 <code>canvas:delegate-mixin</code>	11
6.7 <code>canvas:info<%></code>	11
6.8 <code>canvas:info-mixin</code>	11

6.9	<code>canvas:wide-snip<%></code>	12
6.10	<code>canvas:wide-snip-mixin</code>	13
6.11	<code>canvas:basic% = (canvas:basic-mixin editor-canvas%)</code>	13
6.12	<code>canvas:color% = (canvas:color-mixin canvas:basic%)</code>	14
6.13	<code>canvas:info% = (canvas:info-mixin canvas:basic%)</code>	14
6.14	<code>canvas:delegate% = (canvas:delegate-mixin canvas:basic%)</code>	15
6.15	<code>canvas:wide-snip% = (canvas:wide-snip-mixin canvas:basic%)</code>	16
7	Color	17
7.1	<code>color:text<%></code>	17
7.2	<code>color:text-mixin</code>	21
7.3	<code>color:text% = (color:text-mixin text:keymap%)</code>	23
7.4	<code>color:text-mode<%></code>	24
7.5	<code>color:text-mode-mixin</code>	24
7.6	<code>color:text-mode% = (color:text-mode-mixin mode:surrogate-text%)</code>	24
8	Editor	25
8.1	<code>editor:basic<%></code>	26
8.2	<code>editor:basic-mixin</code>	28
8.3	<code>editor:standard-style-list<%></code>	31
8.4	<code>editor:standard-style-list-mixin</code>	31
8.5	<code>editor:keymap<%></code>	32
8.6	<code>editor:keymap-mixin</code>	32
8.7	<code>editor:autowrap<%></code>	33
8.8	<code>editor:autowrap-mixin</code>	33
8.9	<code>editor:file<%></code>	33
8.10	<code>editor:file-mixin</code>	34
8.11	<code>editor:backup-autosave<%></code>	35
8.12	<code>editor:backup-autosave-mixin</code>	36
8.13	<code>editor:info<%></code>	37

8.14	<code>editor:info-mixin</code>	37
9	Exit	39
10	Exceptions	40
11	Finder	41
12	Frame	42
12.1	<code>frame:basic<%></code>	44
12.2	<code>frame:basic-mixin</code>	46
12.3	<code>frame:size-pref<%></code>	49
12.4	<code>frame:size-pref-mixin</code>	49
12.5	<code>frame:register-group<%></code>	50
12.6	<code>frame:register-group-mixin</code>	50
12.7	<code>frame:status-line<%></code>	51
12.8	<code>frame:status-line-mixin</code>	52
12.9	<code>frame:info<%></code>	53
12.10	<code>frame:info-mixin</code>	55
12.11	<code>frame:text-info<%></code>	56
12.12	<code>frame:text-info-mixin</code>	57
12.13	<code>frame:pasteboard-info<%></code>	58
12.14	<code>frame:pasteboard-info-mixin</code>	58
12.15	<code>frame:standard-menus<%></code>	58
12.16	<code>frame:standard-menus-mixin</code>	91
12.17	<code>frame:editor<%></code>	92
12.18	<code>frame:editor-mixin</code>	94
12.19	<code>frame:open-here<%></code>	99
12.20	<code>frame:open-here-mixin</code>	99
12.21	<code>frame:text<%></code>	101
12.22	<code>frame:text-mixin</code>	101

12.23	<code>frame:pasteboard<%></code>	102
12.24	<code>frame:pasteboard-mixin</code>	102
12.25	<code>frame:delegate<%></code>	103
12.26	<code>frame:delegate-mixin</code>	104
12.27	<code>frame:searchable<%></code>	105
12.28	<code>frame:searchable-mixin</code>	108
12.29	<code>frame:searchable-text<%></code>	111
12.30	<code>frame:searchable-text-mixin</code>	111
12.31	<code>frame:basic% = (frame:register-group-mixin (frame:basic-mixin frame%))</code>	112
12.32	<code>frame:size-pref% = (frame:size-pref-mixin frame:basic%)</code>	112
12.33	<code>frame:info% = (frame:info-mixin frame:basic%)</code>	113
12.34	<code>frame:text-info% = (frame:text-info-mixin frame:info%)</code>	113
12.35	<code>frame:pasteboard-info% = (frame:pasteboard-info-mixin frame:text-info%)</code>	114
12.36	<code>frame:status-line% = (frame:status-line-mixin frame:text-info%)</code>	114
12.37	<code>frame:standard-menus% = (frame:standard-menus-mixin frame:status-line%)</code>	115
12.38	<code>frame:editor% = (frame:editor-mixin frame:standard-menus%)</code>	116
12.39	<code>frame:open-here% = (frame:open-here-mixin frame:editor%)</code>	116
12.40	<code>frame:text% = (frame:text-mixin frame:open-here%)</code>	117
12.41	<code>frame:searchable% = (frame:searchable-mixin (frame:searchable-text-mixin frame:text%))</code>	117
12.42	<code>frame:delegate% = (frame:delegate-mixin frame:searchable%)</code>	118
12.43	<code>frame:pasteboard% = (frame:pasteboard-mixin frame:open-here%)</code>	118
13	Group	120
13.1	<code>group:%</code>	120
14	Handler	124
15	Icon	125
16	Keymap	126
16.1	<code>keymap:aug-keymap<%></code>	126

16.2	<code>keymap:aug-keymap-mixin</code>	127
16.3	<code>keymap:aug-keymap% = (keymap:aug-keymap-mixin keymap%)</code>	128
17	Main	129
18	Menu	130
18.1	<code>menu:can-restore<%></code>	130
18.2	<code>menu:can-restore-mixin</code>	130
18.3	<code>menu:can-restore-underscore<%></code>	131
18.4	<code>menu:can-restore-underscore-mixin</code>	131
18.5	<code>menu:can-restore-menu-item% = (menu:can-restore-mixin menu-item%)</code>	131
18.6	<code>menu:can-restore-checkable-menu-item% = (menu:can-restore-mixin checkable-menu-item%)</code>	131
18.7	<code>menu:can-restore-underscore-menu% = (menu:can-restore-underscore-mixin menu%)</code>	132
19	Mode	133
19.1	<code>mode:host-text<%></code>	133
19.2	<code>mode:surrogate-text<%></code>	133
19.3	<code>mode:host-text-mixin</code>	142
19.4	<code>mode:surrogate-text%</code>	156
20	Panel	166
20.1	<code>panel:single<%></code>	167
20.2	<code>panel:single-mixin</code>	167
20.3	<code>panel:single-window<%></code>	168
20.4	<code>panel:single-window-mixin</code>	168
20.5	<code>panel:single% = (panel:single-mixin (panel:single-window-mixin panel%))</code>	169
20.6	<code>panel:single-pane% = (panel:single-mixin pane%)</code>	169
20.7	<code>panel:dragable<%></code>	170
20.8	<code>panel:vertical-dragable<%></code>	171
20.9	<code>panel:horizontal-dragable<%></code>	171
20.10	<code>panel:dragable-mixin</code>	171

20.11	<code>panel:vertical-dragable-mixin</code>	173
20.12	<code>panel:horizontal-dragable-mixin</code>	173
20.13	<code>panel:vertical-dragable% = (panel:vertical-dragable-mixin (panel:dragable-mixin vertic</code>	
20.14	<code>panel:horizontal-dragable% = (panel:horizontal-dragable-mixin (panel:dragable-mixin h</code>	
21	Pasteboard	175
21.1	<code>pasteboard:basic% = (editor:basic-mixin pasteboard%)</code>	175
21.2	<code>pasteboard:standard-style-list% = (editor:standard-style-list-mixin pasteboard:basic%</code>	
21.3	<code>pasteboard:keymap% = (editor:keymap-mixin pasteboard:standard-style-list%)</code>	175
21.4	<code>pasteboard:file% = (editor:file-mixin pasteboard:keymap%)</code>	176
21.5	<code>pasteboard:backup-autosave% = (editor:backup-autosave-mixin pasteboard:file%)</code>	176
21.6	<code>pasteboard:info% = (editor:info-mixin pasteboard:backup-autosave%)</code> . . .	176
22	Pathname Utilities	177
23	Preferences	178
24	Scheme	179
24.1	<code>scheme:sexp-snip<%></code>	179
24.2	<code>scheme:sexp-snip%</code>	179
24.3	<code>scheme:text<%></code>	182
24.4	<code>scheme:text-mixin</code>	187
24.5	<code>scheme:text-mode<%></code>	188
24.6	<code>scheme:text-mode-mixin</code>	188
24.7	<code>scheme:set-mode-mixin</code>	189
24.8	<code>text-mode%</code>	189
24.9	<code>scheme:text% = (mode:host-text-mixin (scheme:set-mode-mixin (scheme:text-mixin color:</code>	
24.10	<code>scheme:text-mode% = (scheme:text-mode-mixin color:text-mode%)</code>	189
25	Text	191
25.1	<code>text:basic<%></code>	193
25.2	<code>text:basic-mixin</code>	195

25.3	<code>text:foreground-color<%></code>	196
25.4	<code>text:foreground-color-mixin</code>	197
25.5	<code>text:hide-caret/selection<%></code>	197
25.6	<code>text:hide-caret/selection-mixin</code>	197
25.7	<code>text:nbsp->space<%></code>	198
25.8	<code>text:nbsp->space-mixin</code>	198
25.9	<code>text:searching<%></code>	199
25.10	<code>text:searching-mixin</code>	200
25.11	<code>text:return<%></code>	200
25.12	<code>text:return-mixin</code>	200
25.13	<code>text:wide-snip<%></code>	201
25.14	<code>text:wide-snip-mixin</code>	201
25.15	<code>text:delegate<%></code>	202
25.16	<code>text:1-pixel-string-snip%</code>	202
25.17	<code>text:1-pixel-tab-snip%</code>	205
25.18	<code>text:delegate-mixin</code>	208
25.19	<code>text:info<%></code>	211
25.20	<code>text:info-mixin</code>	211
25.21	<code>text:clever-file-format<%></code>	213
25.22	<code>text:clever-file-format-mixin</code>	213
25.23	<code>text:file<%></code>	214
25.24	<code>text:file-mixin</code>	215
25.25	<code>text:ports<%></code>	216
25.26	<code>text:ports-mixin</code>	222
25.27	<code>text:input-box<%></code>	223
25.28	<code>text:ports%</code>	223
25.29	<code>text:input-box-mixin</code>	223
25.30	<code>text:basic% = (editor:basic-mixin (text:basic-mixin text%))</code>	224
25.31	<code>text:hide-caret/selection% = (text:hide-caret/selection-mixin text:basic%)</code>	224

25.32	<code>text:nbsp->space%</code> = (<code>text:nbsp->space-mixin</code> <code>text:basic%</code>)	225
25.33	<code>text:delegate%</code> = (<code>text:delegate-mixin</code> <code>text:basic%</code>)	225
25.34	<code>text:wide-snip%</code> = (<code>text:wide-snip-mixin</code> <code>text:basic%</code>)	225
25.35	<code>text:standard-style-list%</code> = (<code>editor:standard-style-list-mixin</code> <code>text:wide-snip%</code>)	225
25.36	<code>text:input-box%</code> = (<code>text:input-box-mixin</code> <code>text:standard-style-list%</code>) . . .	225
25.37	<code>text:keymap%</code> = (<code>editor:keymap-mixin</code> <code>text:standard-style-list%</code>)	226
25.38	<code>text:return%</code> = (<code>text:return-mixin</code> <code>text:keymap%</code>)	226
25.39	<code>text:autowrap%</code> = (<code>editor:autowrap-mixin</code> <code>text:keymap%</code>)	226
25.40	<code>text:file%</code> = (<code>text:file-mixin</code> (<code>editor:file-mixin</code> <code>text:autowrap%</code>)) . . .	226
25.41	<code>text:clever-file-format%</code> = (<code>text:clever-file-format-mixin</code> <code>text:file%</code>)	227
25.42	<code>text:backup-autosave%</code> = (<code>editor:backup-autosave-mixin</code> <code>text:clever-file-format%</code>)	227
25.43	<code>text:searching%</code> = (<code>text:searching-mixin</code> <code>text:backup-autosave%</code>)	227
25.44	<code>text:info%</code> = (<code>editor:info-mixin</code> (<code>text:info-mixin</code> <code>text:searching%</code>)) . . .	228
26	Version	229
27	Framework Functions	230
27.1	Framework Functions	230
	Index	266

1. This Manual

This manual describes a framework for programmers developing applications MrEd. It assumes familiarity with MrEd as described in *PLT MrEd: Graphical Toolbox Manual* and MzScheme as described in *PLT MzScheme: Language Manual*.

[build date: January 12, 2006]

1.1 Thanks

Thanks to Shriram Krishnamurthi, Cormac Flanagan, Matthias Felleisen, Ian Barland, Gann Bierner, Richard Cobbe, Dan Grossman, Stephanie Weirich, Paul Steckler, Sebastian Good, Johnathan Franklin, Mark Krentel, Corky Cartwright, Michael Ernst, Kennis Koldewyn, Bruce Duba, and many others for their feedback and help.

This manual was typeset using \LaTeX , \SIATeX , and \tex2page . Some typesetting macros were originally taken from Julian Smart's *Reference Manual for wxWindows 1.60: a portable C++ GUI toolkit*.

This manual was typeset on January 12, 2006.

2. Overview

The Framework is a library that provides application framework for MrEd. It is designed to make implementation of an application in MrEd simpler and easier. It provides standard classes and utilities for managing

- frames, §12,
- editors, §8,
- and many others.

See section 3.1 for information on how to load the framework into an application.

3. Preliminaries

3.1 Libraries

The framework provides these libraries:

- **Entire Framework**

- `(require (lib "framework.ss" "framework"))`
This library provides all of the definitions and syntax described in this manual.
- `(require (lib "framework-sig.ss" "framework"))`
This library provides the signature definitions: `#1^`, and `#1^`. The `framework^` signature contains all of the names of the procedures described in this manual, except those that begin with `test:` and `gui-utils:`. The `framework-class^` signature contains all of the classes defined in this manual.
- `(require (lib "framework-unit.ss" "framework"))`
This library provides one `unit/sigs, §` in *PLT MzLib: Libraries Manual*: #1@. It exports the signature `framework^`. It imports the `mred^` signature.

- **Test Suite Engine**

- `(require (lib "test.ss" "framework"))`
This library provides all of the definitions beginning with `test:` described in this manual.

- **GUI Utilities** `(require (lib "gui-utils.ss" "framework"))`

This libraries provides all of the definitions beginning with `gui-utils:` described in this manual.

3.2 Mixins

The framework relies heavily on mixins. A mixin is a class parameterization modeled on a paper published by Flatt, Felleisen, and Krishnamurthi, available at <http://www.cs.utah.edu/plt/publications/icfp98-ff/>. The implementation of these mixins in MzScheme is with the combination of `lambda` and `class`. The framework provides a macro to simplify the checking and implementation of these mixins. Its syntax is very similar to the syntax for `class*`, § in *PLT MzLib: Libraries Manual*. The shape of a mixin is:

```
(mixin (interface-expr ...) (interface-expr ...)
      instance-variable-clause ...)
```

This macro expands into a procedure that accepts a class. The argument passed to this procedure must match the interfaces of the first `interface-exprs` expressions. The procedure returns a class that is derived from its argument. This result class must match the interfaces specified in the second `interface-exprs` section; it has clauses specified by `instance-variable-clauses`. The syntax of the `initialization-variables` and `instance-variable-clause` are exactly the same as `class*/names, §` in *PLT MzLib: Libraries Manual*.

The mixin macro does some checking to be sure that variables that the `instance-variable-clauses` refer to in their super class are in the interfaces. That checking and the checking that the input class matches the declared interfaces aside, the mixin macro's expansion is something like this:

```
(mixin (i<%> ...) (j<%> ...)
  clause ...)
=
(lambda (%)
  (class* % (j<%> ...)
    clause ...))
```

The `i<%>` interfaces do not appear in the output because they are only used for the error checking and are discarded by the time the class is created.

The `mixin` macro is provided by

```
(require (lib "macro.ss" "framework"))
```

3.3 GUI Test Suite Utilities

The framework provides several new primitive functions that simulate user actions, which may be used to test applications. You use these primitives and combine them just as regular MzScheme functions. For example,

```
(begin
  (test:keystroke #\A)
  (test:menu-select "File" "Save"))
```

sends a keystroke event to the window with the keyboard focus and invokes the callback function for the “Save” menu item from the “File” menu. This has the same effect as if the user typed the key “A”, pulled down the “File” menu and selected “Save”.

It is possible to load this portion of the framework without loading the rest of the framework. See [fw:libraries](#) for more details.

Currently, the test engine has primitives for pushing buttons, setting check-boxes and choices, sending keystrokes, selecting menu items and clicking the mouse. Many functions that are also useful in application testing, such as traversing a tree of panels, getting the text from a canvas, determining if a window is shown, and so on, exist in MrEd.

3.3.1 Actions and completeness

The actions associated with a testing primitive may not have finished when the primitive returns to its caller. Some actions may yield control before they can complete. For example, selecting “Save As...” from the “File” menu opens a dialog box and will not complete until the “OK” or “Cancel” button is pushed.

However, all testing functions wait at least a minimum interval before returning to give the action a chance to finish. This interval controls the speed at which the test suite runs, and gives some slack time for events to complete. The default interval is 100 milliseconds. The interval can be queried or set with `test:run-interval`.

A primitive action will not return until the run-interval has expired and the action has finished, raised an error, or yielded. The number of incomplete actions is given by `test:number-pending-actions`.

Note: Once a primitive action is started, it is not possible to undo it or kill its remaining effect. Thus, it is not possible to write a utility that flushes the incomplete actions and resets `number-pending-actions` to zero.

However, actions which do not complete right away often provide a way to cancel themselves. For example, many dialog boxes have a “Cancel” button which will terminate the action with no further effect. But this is accomplished by sending an additional action (the button push), not by undoing the original action.

3.3.2 Errors

Errors in the primitive actions (which necessarily run in the handler thread) are caught and reraised in the calling thread.

However, the primitive actions can only guarantee that the action has started, and they may return before the action has completed. As a consequence, an action may raise an error long after the function that started it has returned. In this case, the error is saved and reraised at the first opportunity (the next primitive action).

The test engine keeps a buffer for one error, saving only the first error. Any subsequent errors are discarded. Reraising an error empties the buffer, allowing the next error to be saved.

The function `test:reraise-error` reraises any pending errors.

3.3.3 Technical Issues

Active Frame

The Self Test primitive actions all implicitly apply to the top-most (active) frame.

Thread Issues

The code started by the primitive actions must run in the handler thread of the eventspace where the event takes place. As a result, the test suite that invokes the primitive actions must *not* run in that handler thread (or else some actions will deadlock). See the eventspace section, §2.4 in *PLT MrEd: Graphical Toolbox Manual* for more info.

Window Manager (Unix only)

In order for the Self Tester to work correctly, the window manager must set the keyboard focus to follow the active frame. This is the default behavior in Microsoft Windows and MacOS, but not in X windows.

In X windows, you must explicitly tell your window manager to set the keyboard focus to the top-most frame, regardless of the position of the actual mouse. Some window managers may not implement such functionality. You can obtain such an effect in Fvwm and Fvwm95 by using the option:

```
Style "*" ClickToFocus
```

4. Application

The procedure in this chapter is used to supply information about your application to the framework.

5. Autosave

Autosaving in MrEd is performed by registering with a single autosaver daemon. Objects that are registered with the autosaver must have a `do-autosave` method that is called periodically with no arguments.

- `autosave:autosavable<%>`

5.1 `autosave:autosavable<%>`

Classes that implement this interface can be autosaved.

`do-autosave`

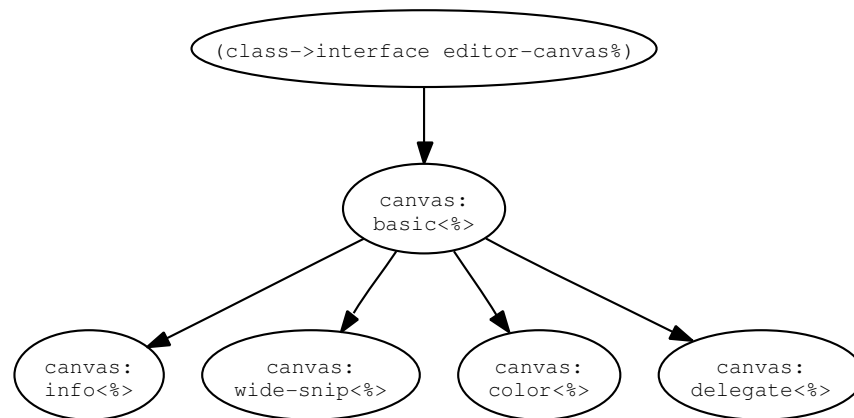
This method is called when the object is registered to be autosaved (see `autosave:register`).

- (send *an-autosave:autosavable* do-autosave) ⇒ void

6. Canvas

This chapter describes the editor canvas mixins, interfaces, and classes.

This is the interface hierarchy:



- `canvas:basic%`
- `canvas:color%`
- `canvas:delegate%`
- `canvas:info%`
- `canvas:wide-snip%`
- `canvas:basic<%>`
- `canvas:color<%>`
- `canvas:delegate<%>`
- `canvas:info<%>`
- `canvas:wide-snip<%>`
- `canvas:basic-mixin`
- `canvas:color-mixin`
- `canvas:delegate-mixin`
- `canvas:info-mixin`
- `canvas:wide-snip-mixin`

6.1 canvas:basic<%>

Extends: (class->interface `editor-canvas%`)

6.2 canvas:basic-mixin

Domain: (class->interface `editor-canvas%`)

Implements: `canvas:basic<%>`

```

- init args: (parent _) [(editor _)] [(style _)] [(scrolls-per-page _)] [(label
_) [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
_) [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
_) [(stretchable-width _)] [(stretchable-height _)]
  parent: frame%, dialog%, panel%, or pane% object
  editor = #f: text% or pasteboard% object or #f
  style = null: list of symbols in '(no-border control-border combo no-hscroll
no-vscroll hide-hscroll hide-vscroll auto-vscroll
auto-hscroll resize-corner deleted transparent)
  scrolls-per-page = 100: exact integer in [1, 10000]
  label = #f: string (up to 200 characters) or #f
  wheel-step = 3: exact integer in [1, 10000] or #f
  line-count = #f: exact integer in [1, 1000] or #f
  horizontal-inset = 5: exact integer in [0, 1000]
  vertical-inset = 5: exact integer in [0, 1000]
  enabled = #t: boolean
  vert-margin = 0: exact integer in [0, 1000]
  horiz-margin = 0: exact integer in [0, 1000]
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean

```

If a canvas is initialized with #f for `editor`, install an editor later with `set-editor`.

The `style` list can contain the following flags:

- 'no-border — omits a border around the canvas
- 'control-border — gives the canvas a border that is like a `text-field%` control
- 'combo — gives the canvas a combo button that is like a `combo-field%` control; this style is intended for use with 'control-border, 'hide-hscroll, and 'hide-vscroll
- 'no-hscroll — disallows horizontal scrolling and hides the horizontal scrollbar
- 'no-vscroll — disallows vertical scrolling and hides the vertical scrollbar
- 'hide-hscroll — allows horizontal scrolling, but hides the horizontal scrollbar
- 'hide-vscroll — allows vertical scrolling, but hides the vertical scrollbar
- 'auto-hscroll — automatically hides the horizontal scrollbar when unneeded (unless 'no-hscroll or 'hide-hscroll is specified)

- 'auto-vscroll — automatically hides the vertical scrollbar when unneeded (unless 'no-vscroll or 'hide-vscroll is specified)
- 'resize-corner — leaves room for a resize control at the canvas's bottom right when only one scrollbar is visible
- 'deleted — creates the canvas as initially hidden and without affecting *parent*'s geometry; the canvas can be made active later by calling *parent*'s `add-child` method
- 'transparent — the canvas is “erased” before an update using it's parent window's background

While vertical scrolling of text editors is based on lines, horizontal scrolling and pasteboard vertical scrolling is based on a fixed number of steps per horizontal page. The `scrolls-per-page` argument sets this value.

If provided, the `wheel-step` argument is passed on to the `wheel-step` method. The default wheel step can be overridden globally though the '|MrEd:wheelStep| preference; see “Preferences” (§12 in *PLT MrEd: Graphical Toolbox Manual*).

If `line-count` is not #f, it is passed on to the `set-line-count` method.

If `horizontal-inset` is not 5, it is passed on to the `horizontal-inset` method. Similarly, if `vertical-inset` is not 5, it is passed on to the `vertical-inset` method.

For information about the `enabled` argument, see `window<%>`. For information about the `horiz-margin` and `vert-margin` arguments, see `subarea<%>`. For information about the `min-width`, `min-height`, `stretchable-width`, and `stretchable-height` arguments, see `area<%>`.

6.3 canvas:color<%>

Extends: `canvas:basic<%>`

Mixins that implement this interface initialize the background color of the canvas to the value of the 'framework:basic-canvas-background preference. Adds a callback so that when that preference is modified, the background color changes.

6.4 canvas:color-mixin

Domain: `canvas:basic<%>`

Implements: `canvas:color<%>`

Implements: `canvas:basic<%>`

6.5 canvas:delegate<%>

Extends: `canvas:basic<%>`

This class is part of the delegate window implementation.

6.6 canvas:delegate-mixin

Domain: `canvas:basic<%>`

Implements: `canvas:basic<%>`

Implements: `canvas:delegate<%>`

Provides an implementation of `canvas:delegate<%>`.

`on-superwindow-show`

Called via the event queue whenever the visibility of a window has changed, either through a call to the window's `show`, through the showing/hiding of one of the window's ancestors, or through the activating or deactivating of the window or its ancestor in a container (e.g., via `delete-child`). The method's argument indicates whether the window is now visible or not.

This method is not called when the window is initially created; it is called only after a change from the window's initial visibility. Furthermore, if a show notification event is queued for the window and it reverts its visibility before the event is dispatched, then the dispatch is canceled.

```
- (send a-canvas:delegate-mixin on-superwindow-show shown? void
  shown? : boolean
```

Notifies the delegate window when the original window is visible. When invisible, the blue highlighting is erased.

6.7 canvas:info<%>

Extends: `canvas:basic<%>`

6.8 canvas:info-mixin

Domain: `canvas:basic<%>`

Implements: `canvas:info<%>`

Implements: `canvas:basic<%>`

`on-focus`

Called when a window receives or loses the keyboard focus. If the argument is `#t`, the keyboard focus was received, otherwise it was lost.

Note that under X, keyboard focus can move to the menu bar when the user is selecting a menu item.

```
- (send a-canvas:info-mixin on-focus void
  sets the canvas that the frame displays info about.
```

`set-editor`

Sets the editor that is displayed by the canvas, releasing the current editor (if any). If the new editor already has an administrator that is not associated with an `editor-canvas%`, then the new editor is *not* installed into the canvas.

```
- (send a-canvas:info-mixin set-editor void
  Calls update-info to update the frame's info panel.
```

6.9 canvas:wide-snip<%>

Extends: `canvas:basic<%>`

Any `canvas%` that matches this interface will automatically resize selected snips when it's size changes. Use `add-tall-snip` and `add-wide-snip` to specify which snips should be resized.

`add-tall-snip`

Snips passed to this method will be resized when the canvas's size changes. Their height will be set so they take up all of the space from their tops to the bottom of the canvas.

```
- (send a-canvas:wide-snip add-tall-snip snip) => void
  snip: (instance snip%)
```

`add-wide-snip`

Snips passed to this method will be resized when the canvas's size changes. Their width will be set so they take up all of the space from their lefts to the right edge of the canvas.

```
- (send a-canvas:wide-snip add-wide-snip snip) ⇒ void
   snip: (instance snip%)
```

recalc-snips

Recalculates the sizes of the wide snips.

```
- (send a-canvas:wide-snip recalc-snips) ⇒ void
```

6.10 canvas:wide-snip-mixin

Domain: `canvas:basic<%>`

Implements: `canvas:basic<%>`

Implements: `canvas:wide-snip<%>`

This canvas maintains a list of wide and tall snips and adjusts their heights and widths when the canvas's size changes.

The result of this mixin uses the same initialization arguments as the mixin's argument.

on-size

Called when the window is resized. The window's new size (in pixels) is provided to the method. The size values are for the entire window, not just the client area.

```
- (send a-canvas:wide-snip-mixin on-size width height void
   width: exact integer in [0, 10000]
   height: exact integer in [0, 10000])
```

Adjusts the sizes of the marked snips.

See `add-wide-snip` and `add-tall-snip`.

6.11 canvas:basic% = (canvas:basic-mixin editor-canvas%)

```
canvas:basic% = (canvas:basic-mixin editor-canvas%)
```

```
- (new canvas:basic% (parent _) [(editor _)] [(style _)] [(scrolls-per-page _)]
  [(label _)] [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
  _)] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
  _)] [(stretchable-width _)] [(stretchable-height _)]) ⇒ canvas:basic% object
   parent:
     frame%, dialog%, panel%, or pane% object
```

```

editor = #f: text% or pasteboard% object or #f
style = null: list of symbols in '(no-border control-border combo no-hscroll
      no-vscroll hide-hscroll hide-vscroll auto-vscroll
      auto-hscroll resize-corner deleted transparent)
scrolls-per-page = 100: exact integer in [1, 10000]
label = #f: string (up to 200 characters) or #f
wheel-step = 3: exact integer in [1, 10000] or #f
line-count = #f: exact integer in [1, 1000] or #f
horizontal-inset = 5: exact integer in [0, 1000]
vertical-inset = 5: exact integer in [0, 1000]
enabled = #t: boolean
vert-margin = 0: exact integer in [0, 1000]
horiz-margin = 0: exact integer in [0, 1000]
min-width = 0: exact integer in [0, 10000]
min-height = 0: exact integer in [0, 10000]
stretchable-width = #t: boolean
stretchable-height = #t: boolean

```

Passes all arguments to super-init.

6.12 canvas:color% = (canvas:color-mixin canvas:basic%)

```
canvas:color% = (canvas:color-mixin canvas:basic%)
```

```

- (new canvas:color% (parent _) [(editor _)] [(style _)] [(scrolls-per-page _)]
  [(label _)] [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
  _)] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
  _)] [(stretchable-width _)] [(stretchable-height _)]) => canvas:color% object
parent:
  frame%, dialog%, panel%, or pane% object
editor = #f: text% or pasteboard% object or #f
style = null: list of symbols in '(no-border control-border combo no-hscroll
      no-vscroll hide-hscroll hide-vscroll auto-vscroll
      auto-hscroll resize-corner deleted transparent)
scrolls-per-page = 100: exact integer in [1, 10000]
label = #f: string (up to 200 characters) or #f
wheel-step = 3: exact integer in [1, 10000] or #f
line-count = #f: exact integer in [1, 1000] or #f
horizontal-inset = 5: exact integer in [0, 1000]
vertical-inset = 5: exact integer in [0, 1000]
enabled = #t: boolean
vert-margin = 0: exact integer in [0, 1000]
horiz-margin = 0: exact integer in [0, 1000]
min-width = 0: exact integer in [0, 10000]
min-height = 0: exact integer in [0, 10000]
stretchable-width = #t: boolean
stretchable-height = #t: boolean

```

Passes all arguments to super-init.

6.13 canvas:info% = (canvas:info-mixin canvas:basic%)

```
canvas:info% = (canvas:info-mixin canvas:basic%)
```

```
- (new canvas:info% (parent _) [(editor _)] [(style _)] [(scrolls-per-page _)]
  [(label _)] [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
  _)] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
  _)] [(stretchable-width _)] [(stretchable-height _)]) => canvas:info% object
  parent: frame%, dialog%, panel%, or pane% object
  editor = #f: text% or pasteboard% object or #f
  style = null: list of symbols in '(no-border control-border combo no-hscroll
    no-vscroll hide-hscroll hide-vscroll auto-vscroll
    auto-hscroll resize-corner deleted transparent)
  scrolls-per-page = 100: exact integer in [1, 10000]
  label = #f: string (up to 200 characters) or #f
  wheel-step = 3: exact integer in [1, 10000] or #f
  line-count = #f: exact integer in [1, 1000] or #f
  horizontal-inset = 5: exact integer in [0, 1000]
  vertical-inset = 5: exact integer in [0, 1000]
  enabled = #t: boolean
  vert-margin = 0: exact integer in [0, 1000]
  horiz-margin = 0: exact integer in [0, 1000]
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to super-init.

6.14 canvas:delegate% = (canvas:delegate-mixin canvas:basic%)

```
canvas:delegate% = (canvas:delegate-mixin canvas:basic%)
```

```
- (new canvas:delegate% (parent _) [(editor _)] [(style _)] [(scrolls-per-page
  _)] [(label _)] [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
  _)] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
  _)] [(stretchable-width _)] [(stretchable-height _)]) => canvas:delegate% ob-
  ject
  parent: frame%, dialog%, panel%, or pane% object
  editor = #f: text% or pasteboard% object or #f
  style = null: list of symbols in '(no-border control-border combo no-hscroll
    no-vscroll hide-hscroll hide-vscroll auto-vscroll
    auto-hscroll resize-corner deleted transparent)
  scrolls-per-page = 100: exact integer in [1, 10000]
  label = #f: string (up to 200 characters) or #f
  wheel-step = 3: exact integer in [1, 10000] or #f
  line-count = #f: exact integer in [1, 1000] or #f
  horizontal-inset = 5: exact integer in [0, 1000]
  vertical-inset = 5: exact integer in [0, 1000]
  enabled = #t: boolean
  vert-margin = 0: exact integer in [0, 1000]
  horiz-margin = 0: exact integer in [0, 1000]
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to super-init.

6.15 canvas:wide-snip% = (canvas:wide-snip-mixin canvas:basic%)

```
canvas:wide-snip% = (canvas:wide-snip-mixin canvas:basic%)
```

```
- (new canvas:wide-snip% (parent _) [(editor _)] [(style _)] [(scrolls-per-page
_) ] [(label _)] [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
_) ] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
_) ] [(stretchable-width _)] [(stretchable-height _)]) => canvas:wide-snip% ob-
ject
  parent:      frame%, dialog%, panel%, or pane% object
  editor = #f:  text% or pasteboard% object or #f
  style = null: list of symbols in '(no-border control-border combo no-hscroll
          no-vscroll hide-hscroll hide-vscroll auto-vscroll
          auto-hscroll resize-corner deleted transparent)
  scrolls-per-page = 100: exact integer in [1, 10000]
  label = #f:   string (up to 200 characters) or #f
  wheel-step = 3: exact integer in [1, 10000] or #f
  line-count = #f: exact integer in [1, 1000] or #f
  horizontal-inset = 5: exact integer in [0, 1000]
  vertical-inset = 5: exact integer in [0, 1000]
  enabled = #t:  boolean
  vert-margin = 0: exact integer in [0, 1000]
  horiz-margin = 0: exact integer in [0, 1000]
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to super-init.

7. Color

- `color:text-mode%`
- `color:text%`
- `color:text-mode<%>`
- `color:text<%>`
- `color:text-mixin`
- `color:text-mode-mixin`

7.1 `color:text<%>`

Extends: `text:basic<%>`

This interface describes how coloring is stopped and started for text that knows how to color itself. It also describes how to query the lexical and s-expression structure of the text.

`backward-containing-sexp`

- `(send a-color:text backward-containing-sexp) ⇒ void`
- `(send a-color:text backward-containing-sexp position cutoff) ⇒ (union natural-number? false?)`
 `position: natural-number?`
 `cutoff: natural-number?`

Return the starting position of the interior of the (non-atomic) s-expression containing position, or #f if there is none.

Must only be called while the tokenizer is started.

`backward-match`

- `(send a-color:text backward-match position cutoff) ⇒ (union natural-number? false?)`
 `position: natural-number?`
 `cutoff: natural-number?`

Skip all consecutive whitespaces and comments (using `skip-whitespace`) immediately preceding the position. If the token at this position is a close, return the position of the matching open, or `#f` if there is none. If the token was an open, return `#f`. For any other token, return the start of that token.

Must only be called while the tokenizer is started.

`classify-position`

```
- (send a-color:text classify-position position) ⇒ symbol
   position: natural-number?
```

Return a symbol for the lexer-determined token type for the token that contains the item after `position`.

Must only be called while the tokenizer is started.

`force-stop-colorer`

Causes the entire tokenizing/coloring system to become inactive. Intended for debugging purposes only.

```
- (send a-color:text force-stop-colorer stop?) ⇒ void
   stop?: boolean
```

`stop?` determines whether the system is being forced to stop or allowed to wake back up.

`forward-match`

```
- (send a-color:text forward-match position cutoff) ⇒ (union natural-number? false?)
   position: natural-number?
   cutoff: natural-number?
```

Skip all consecutive whitespaces and comments (using `skip-whitespace`) immediately following position. If the token at this position is an open, return the position of the matching close, or `#f` if there is none. For any other token, return the end of that token.

Must only be called while the tokenizer is started.

`freeze-colorer`

Keep the text tokenized and paren matched, but stop altering the colors.

```
- (send a-color:text freeze-colorer) ⇒ void
```

`freeze-colorer` will not return until the coloring/tokenization of the entire text is brought up-to-date. It must not be called on a locked text.

insert-close-paren

```
- (send a-color:text insert-close-paren position char flash? fixup?) ⇒ void
  position: natural-number?
  char: char?
  flash?: boolean
  fixup?: boolean
```

Position is the place to put the parenthesis and char is the parenthesis to be added. If fixup? is true, the right kind of closing parenthesis will be chosen from the pairs list kept last passed to start-colorer, otherwise char will be inserted, even if it is not the right kind. If flash? is true the matching open parenthesis will be flashed.

is-frozen?

Indicates if this editor's colorer is frozen. See also `freeze-colorer` and `thaw-colorer`.

```
- (send a-color:text is-frozen?) ⇒ boolean
```

is-stopped?

Indicates if the colorer for this editor has been stopped, or not.

```
- (send a-color:text is-stopped?) ⇒ boolean
```

reset-region

Set the region of the text that is tokenized.

```
- (send a-color:text reset-region start end) ⇒ void
  start: natural-number?
  end: (union 'end natural-number?)
```

skip-whitespace

Returns the next non-whitespace character.

```
- (send a-color:text skip-whitespace position direction comments?) ⇒ natural-
  number?
  position: natural-number?
  direction: (symbols 'forward 'backward)
  comments?: boolean
```

Starts from position and skips whitespace in the direction indicated by direction. If comments? is true, comments are skipped as well as whitespace. skip-whitespace determines whitespaces and comments by comparing the token type to 'white-space and 'comment.

Must only be called while the tokenizer is started.

start-colorer

Starts tokenizing the buffer for coloring and parenthesis matching.

```
- (send a-color:text start-colorer token-sym-style get-token pairs) => void
  token-sym-style: (symbol? .-i . string?)
  get-token: (input-port? .-i . (values any? symbol? (union false? symbol?) natural-number? natural-number?))
  pairs: (listof (list/p symbol? symbol?))
```

token-sym-style will be passed the first return symbol from get-token and should return the style-name that the token should be colored.

get-token takes an input port and returns the next token as 5 values:

1. An unused value. This value is intended to represent the textual component of the token and may be used as such in the future.
2. A symbol describing the type of the token. This symbol is transformed into a style-name via the token-sym-*i*-style argument. The symbols 'white-space and 'comment have special meaning and should always be returned for white space and comment tokens respectively. The symbol 'no-color can be used to indicate that although the token is not white space, it should not be colored. The symbol 'eof must be used to indicate when all the tokens have been consumed.
3. A symbol indicating how the token should be treated by the paren matcher or #f. This symbol should be in the pairs argument.
4. The starting position of the token.
5. The ending position of the token.

get-token will usually be implemented with a lexer using the (lib "lex.ss" "parser-tools") library.

get-token must obey the following invariants:

- Every position in the buffer must be accounted for in exactly one token.
- The token returned by get-token must rely only on the contents of the input port argument. This means that the tokenization of some part of the input cannot depend on earlier parts of the input.
- No edit to the buffer can change the tokenization of the buffer prior to the token immediately preceding the edit. In the following example this invariant does not hold. If the buffer contains:

```
" 1 2 3
```

and the tokenizer treats the unmatched " as its own token (a string error token), and separately tokenizes the 1 2 and 3, an edit to make the buffer look like:

```
" 1 2 3"
```

would result in a single string token modifying previous tokens. To handle these situations, get-token must treat the first line as a single token.

pairs is a list of different kinds of matching parens. The second value returned by get-token is compared to this list to see how the paren matcher should treat the token. An example: Suppose pairs is '((((| |))) (| | |)) (begin end)'. This means that there are three kinds of parens. Any token which has 'begin as its second return value will act as an open for matching tokens with 'end. Similarly any token with '| | | will act as a closing match for tokens with '| | |. When trying to correct a mismatched closing parenthesis, each closing symbol in pairs will be converted to a string and tried as a closing parenthesis.

stop-colorer

Stops coloring and paren matching the buffer.

```
- (send a-color:text stop-colorer clear-colors) ⇒ void
  clear-colors = #t : boolean
```

If clear-colors is true all the text in the buffer will have it's style set to Standard.

thaw-colorer

Start coloring a frozen buffer again.

```
- (send a-color:text thaw-colorer recolor retokenize) ⇒ void
  recolor = #t : boolean
  retokenize = #f : boolean
```

If recolor? is #t, the text is re-colored. If it is #f the text is not recolored. When recolor? is #t, retokenize? controls how the text is recolored. #f causes the text to be entirely re-colored before thaw-colorer returns using the existing tokenization. #t causes the entire text to be retokenized and recolored from scratch. This will happen in the background after the call to thaw-colorer returns.

update-region-end

Modifies the end of the region of the text to be tokenized.

```
- (send a-color:text update-region-end end) ⇒ void
  end : natural-number?
```

7.2 color:text-mixin

Domain: `text:basic<%>`

Implements: `text:basic<%>`

Implements: `color:text<%>`

Adds the functionality needed for on-the-fly coloring and parenthesis matching based on incremental tokenization of the text.

after-change-style (*augments, and augmentable only*)

Called after the style is changed for a given range (and after the **display** is refreshed; use **on-change-style** and **begin-edit-sequence** to avoid extra refreshes when **after-change-style** modifies the editor).

See also **can-change-style?** and **on-edit-sequence**.

No internals locks are set when this method is called.

```
- (send a-color:text-mixin after-change-style void
```

after-delete (augments, and augmentable only)

Called after a given range is deleted from the editor (and after the **display** is refreshed; use **on-delete** and **begin-edit-sequence** to avoid extra refreshes when **after-delete** modifies the editor).

See also **can-delete?** and **on-edit-sequence**.

No internals locks are set when this method is called.

```
- (send a-color:text-mixin after-delete void
```

after-edit-sequence (augments, and augmentable only)

Called after a top-level edit sequence completes (involving unnested **begin-edit-sequence** and **end-edit-sequence**).

See also **on-edit-sequence**.

```
- (send a-color:text-mixin after-edit-sequence void
```

after-insert (augments, and augmentable only)

Called after **items** are inserted into the editor (and after the **display** is refreshed; use **on-insert** and **begin-edit-sequence** to avoid extra refreshes when **after-insert** modifies the editor).

See also **can-insert?** and **on-edit-sequence**.

No internals locks are set when this method is called.

```
- (send a-color:text-mixin after-insert void
```

after-set-position (augments, and augmentable only)

Called after the start and end **position** have been moved (but not when the **position** is moved due to inserts or deletes).

See also [on-edit-sequence](#).

```
- (send a-color:text-mixin after-set-position void
```

lock

Locks or unlocks the editor for modifications. If an editor is locked, *all* modifications are blocked, not just user modifications.

See also [is-locked?](#).

This method does not affect internal locks, as discussed in “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*).

```
- (send a-color:text-mixin lock void
```

on-focus

Called when the keyboard focus changes into or out of this editor (and not to/from a snip within the editor) with #t if the focus is being turned on, #f otherwise.

```
- (send a-color:text-mixin on-focus void
```

on-set-size-constraint (*augments, and augmentable only*)

Called before the editor’s maximum or minimum height or width is changed, after [can-set-size-constraint?](#) is called to verify that the change is ok. The [after-set-size-constraint](#) method is guaranteed to be called after the change has completed.

(This callback method is provided because setting an editor’s maximum width may cause lines to be re-flowed with soft carriage returns.)

See also [on-edit-sequence](#).

```
- (send a-color:text-mixin on-set-size-constraint void
```

7.3 color:text% = (color:text-mixin text:keymap%)

```
color:text% = (color:text-mixin text:keymap%)
```

```
- (new color:text% [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]) => color:text%
  object
```

```
  line-spacing = 1.0: non-negative real number
```

```
  tab-stops = null: list of real numbers
```

```
  auto-wrap = #f: boolean
```

Passes all arguments to super-init.

7.4 color:text-mode<%>**7.5** color:text-mode-mixin

Domain: `mode:surrogate-text<%>`

Implements: `color:text-mode<%>`

Implements: `mode:surrogate-text<%>`

on-disable-surrogate

Called when this surrogate is removed from *object*. See also `set-surrogate`.

```
- (send a-color:text-mode-mixin on-disable-surrogate void
```

on-enable-surrogate

Called when the surrogate is enabled, with the object it is enabled in. See also `set-surrogate`.

```
- (send a-color:text-mode-mixin on-enable-surrogate void
```

7.6 color:text-mode% = (color:text-mode-mixin mode:surrogate-text%)

```
color:text-mode% = (color:text-mode-mixin mode:surrogate-text%)
```

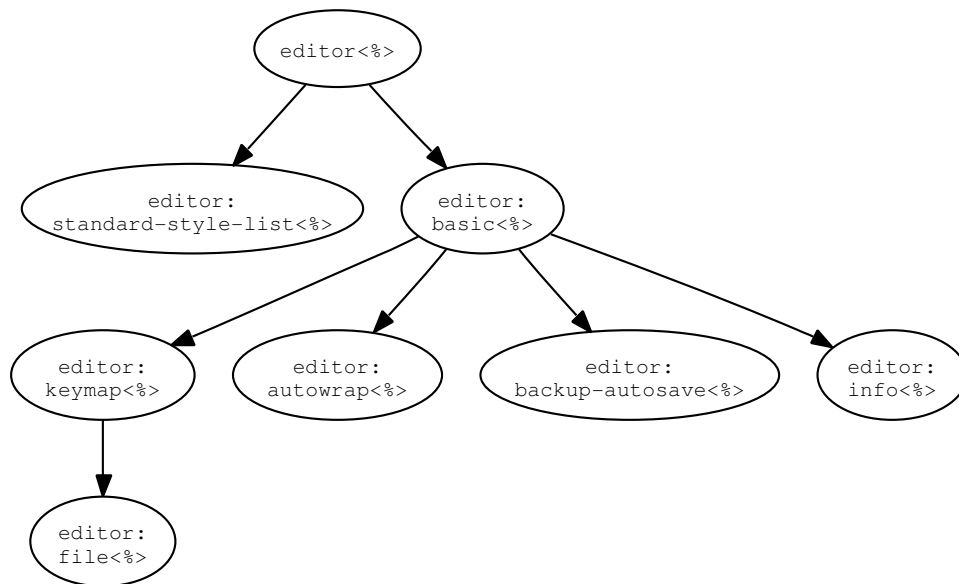
```
- (new color:text-mode% ) => color:text-mode% object
```

Passes all arguments to `super-init`.

8. Editor

This chapter describes the editor mixins, interfaces, and classes.

This is the interface hierarchy for the editor and text classes in the framework:



- `editor:autowrap<%>`
- `editor:backup-autosave<%>`
- `editor:basic<%>`
- `editor:file<%>`
- `editor:info<%>`
- `editor:keymap<%>`
- `editor:standard-style-list<%>`
- `editor:autowrap-mixin`
- `editor:backup-autosave-mixin`
- `editor:basic-mixin`
- `editor:file-mixin`
- `editor:info-mixin`

- editor:keymap-mixin
- editor:standard-style-list-mixin

8.1 editor:basic<%>

Extends: `editor<%>`

Classes matching this interface support the basic `editor<%>` functionality required by the framework.

`can-close?`

This method is called to query the editor if is okay to close the editor. Although there is no visible effect associated with closing an editor, there may be some cleanup actions that need to be run when the user is finished with the editor (asking if it should be saved, for example).

See also `on-close` and `close`.

```
- (send an-editor:basic can-close?) => boolean
```

Returns #t.

`close`

This method is merely

```
(if (can-close?)
    (begin (on-close) #t)
    #f)
```

It is intended as a shorthand, helper method for closing an editor. See also `can-close?` and `on-close`.

```
- (send an-editor:basic close) => boolean
```

`get-filename/untitled-name`

Returns the printed version of the filename for this editor. If the editor doesn't yet have a filename, it returns a symbolic name (something like "Untitled").

```
- (send an-editor:basic get-filename/untitled-name) => string
```

get-top-level-window

Returns the `top-level-window<%>` currently associated with this buffer.

This does not work for embedded editors.

```
- (send an-editor:basic get-top-level-window) ⇒ (union #f (implements top-level-window<%>))
```

has-focus?

This function returns `#t` when the editor has the keyboard focus. It is implemented using: `on-focus`

```
- (send an-editor:basic has-focus?) ⇒ boolean
```

load-file/gui-error

This method is an alternative to `load-file`. Rather than showing errors via the original stdout, it opens a dialog with an error message showing the error.

The result indicates if an error happened (the error has already been shown to the user). It returns `#t` if no error occurred and `#f` if an error occurred.

```
- (send an-editor:basic load-file/gui-error filename format show-errors?) ⇒
boolean
  filename = #f: (union string #f)
  format = 'guess: (union 'guess 'standard 'text 'text-force-cr 'same 'copy)
  show-errors? = #t: boolean
```

local-edit-sequence?

Indicates if this editor is in an edit sequence. Enclosing buffer's edit-sequence status is not considered by this method.

See `begin-edit-sequence` and `end-edit-sequence` for more info about edit sequences.

```
- (send an-editor:basic local-edit-sequence?) ⇒ boolean
```

on-close

This method is called when an editor is closed. See also `can-close?` and `close`.

```
- (send an-editor:basic on-close) ⇒ void
```

Does nothing.

run-after-edit-sequence

This method is used to install callbacks that will be run after any edit-sequence completes.

```
- (send an-editor:basic run-after-edit-sequence thunk tag) => void
  thunk : (-i void)
  tag = #f : (union symbol #f)
```

The procedure *thunk* will be called immediately if the edit is not in an edit-sequence. If the edit is in an edit-sequence, it will be called when the edit-sequence completes.

If *tag* is a symbol, the *thunk* is keyed on that symbol, and only one *thunk* per symbol will be called after the edit-sequence. Specifically, the last call to `run-after-edit-sequence`'s argument will be called.

save-file-out-of-date?

Returns #t if the file on disk has been modified, by some other program.

```
- (send an-editor:basic save-file-out-of-date?) => boolean
```

save-file/gui-error

This method is an alternative to `save-file`. Rather than showing errors via the original stdout, it opens a dialog with an error message showing the error.

The result indicates if an error happened (the error has already been shown to the user). It returns #t if no error occurred and #f if an error occurred.

```
- (send an-editor:basic save-file/gui-error filename format show-errors?) =>
  boolean
  filename = #f : (union string #f)
  format = 'same : (union 'guess 'standard 'text 'text-force-cr same copy)
  show-errors? = #t : boolean
```

8.2 editor:basic-mixin

Domain: `editor<%>`

Implements: `editor<%>`

Implements: `editor:basic<%>`

This provides the basic editor services required by the rest of the framework.

The result of this mixin uses the same initialization arguments as the mixin's argument.

Each instance of a class created with this mixin contains a private `keymap%` that is chained to the global keymap via: `(send keymap chain-to-keymap (keymap:get-global) #f)`.

This installs the global keymap `keymap:get-global` to handle keyboard and mouse mappings not handled by `keymap`. The global keymap is created when the framework is invoked.

`after-edit-sequence` (*augments, and augmentable only*)

Called after a top-level edit sequence completes (involving unnested `begin-edit-sequence` and `end-edit-sequence`).

See also `on-edit-sequence`.

```
- (send an-editor:basic-mixin after-edit-sequence void
  Helps to implement run-after-edit-sequence.
```

`after-load-file` (*augments, and augmentable only*)

Called just after the editor is loaded from a file.

The argument to the method originally specified whether the save was successful, but failures now trigger exceptions such that the method is not even called. Consequently, the argument is always `#t`.

See also `can-load-file?` and `on-load-file`.

```
- (send an-editor:basic-mixin after-load-file success? void
  success?: boolean
  Updates a private instance variable with the modification time of the file, for using in implementing
  save-file-out-of-date?
```

`after-save-file` (*augments, and augmentable only*)

Called just after the editor is saved to a file.

The argument to the method originally specified whether the save was successful, but failures now trigger exceptions such that the method is not even called. Consequently, the argument is always `#t`.

See also `can-save-file?` and `on-save-file`.

```
- (send an-editor:basic-mixin after-save-file success? void
  success?: boolean
  If the current filename is not a temporary filename, this method calls handler:add-to-recent with the
  current filename.
```

to add the new filename to the list of recently opened files.

Additionally, updates a private instance variable with the modification time of the file, for using in implementing `save-file-out-of-date?`.

`can-save-file?` (*augments, and augmentable only*)

Called just before the editor is saved to a file. If the return value is `#f`, the file is not saved. See also `on-save-file` and `after-save-file`.

```
- (send an-editor:basic-mixin can-save-file? filename format boolean
  filename: string
  format: symbol)
```

Checks to see if the file on the disk has been modified out side of this editor, using `save-file-out-of-date?`. If it has, this method prompts the user to be sure they want to save.

`get-file`

Called when the user must be queried for a filename to load an editor. A starting-directory path is passed in, but is may be `#f` to indicate that any directory is fine.

```
- (send an-editor:basic-mixin get-file directory string
  directory: path or #f)
```

Uses `finder:get-file` to find a filename. Also, sets the parameter `finder:dialog-parent-parameter` to the result of `get-top-level-window`.

`on-edit-sequence` (*augments, and augmentable only*)

Called just after a top-level (i.e., unnested) edit sequence starts.

During an edit sequence, all callbacks methods are invoked normally, but it may be appropriate for these callbacks to delay computation during an edit sequence. The callbacks must manage this delay manually. Thus, when overriding other callback methods, such as `on-insert in text%`, `on-insert in pasteboard%`, `after-insert in text%`, or `after-insert in pasteboard%`, consider overriding `on-edit-sequence` and `after-edit-sequence` as well.

“Top-level edit sequence” refers to an outermost pair of `begin-edit-sequence` and `end-edit-sequence` calls. The embedding of an editor within another editor does not affect the timing of calls to `on-edit-sequence`, even if the embedding editor is in an edit sequence.

Pairings of `on-edit-sequence` and `after-edit-sequence` can be nested if an `after-edit-sequence` starts a new edit sequence, since `after-edit-sequence` is called after an edit sequence ends. However, `on-edit-sequence` can never start a new top-level edit sequence (except through an unpaired `end-edit-sequence`), because it is called after a top-level edit sequence starts.

```
- (send an-editor:basic-mixin on-edit-sequence boolean)
```

Always returns #t. Updates a flag for `local-edit-sequence?`

on-focus

Called when the keyboard focus changes into or out of this editor (and not to/from a snip within the editor) with #t if the focus is being turned on, #f otherwise.

```
- (send an-editor:basic-mixin on-focus on? void
   on? : boolean)
```

on-new-box

Creates and returns a new snip for an embedded editor. This method is called by `insert-box`.

```
- (send an-editor:basic-mixin on-new-box type (instance editor-snip%)
   type: (union 'pasteboard 'text))
```

Creates instances of `pasteboard:basic%` or `text:basic%` instead of the built in `pasteboard%` and `text%` classes.

put-file

Called when the user must be queried for a filename to save an editor. Starting-directory and default-name paths are passed in, but either may be #f to indicate that any directory is fine or there is no default name.

```
- (send an-editor:basic-mixin put-file directory default-name string
   directory: path or #f
   default-name: path or #f)
```

Uses `finder:put-file` to find a filename. Also, sets the parameter `finder:dialog-parent-parameter` to the result of `get-top-level-window`.

8.3 editor:standard-style-list<%>

Extends: `editor<%>`

This interface is implemented by the results of `editor:standard-style-list-mixin`.

8.4 editor:standard-style-list-mixin

Domain: `editor<%>`

Implements: `editor:standard-style-list<%>`

Implements: `editor<%>`

The mixin adds code to the initialization of the class that sets the editor's style list (via `set-style-list`) to the result of `editor:get-standard-style-list`.

In addition, it calls `set-load-overwrites-styles` with `#f`. This ensures that saved files with different settings for the style list do not clobber the shared style list.

8.5 editor:keymap<%>

Extends: `editor:basic<%>`

Classes matching this interface add support for mixing in multiple keymaps. They provides an extensible interface to chained keymaps, through the `get-keymaps` method.

This editor is initialized by calling `add-editor-keymap-functions`, `add-text-keymap-functions`, and `add-pasteboard-keymap-functions`.

`get-keymaps`

The keymaps returned from this method are chained to this `editor<%>`'s keymap.

```
- (send an-editor:keymap get-keymaps) ⇒ (list-of (instance keymap%))
```

```
  Defaultly returns (list keymap:get-global)
```

8.6 editor:keymap-mixin

Domain: `editor:basic<%>`

Implements: `editor:basic<%>`

Implements: `editor:keymap<%>`

This provides a mixin that implements the `editor:keymap<%>` interface.

8.7 editor:autowrap<%>

Extends: `editor:basic<%>`

Classes implementing this interface keep the the `auto-wrap` state set based on the `'framework:auto-set-wrap?` preference (see [fw:preferences](#)).

They install a preferences callback with `preferences:add-callback` that sets the state when the preference changes and initialize the value of `auto-wrap` to the current value of `'framework:auto-set-wrap?` via `preferences:get`.

8.8 editor:autowrap-mixin

Domain: `editor:basic<%>`

Implements: `editor:autowrap<%>`

Implements: `editor:basic<%>`

See `editor:autowrap<%>`

8.9 editor:file<%>

Extends: `editor:keymap<%>`

Objects supporting this interface are expected to support files.

`allow-close-with-no-filename?`

This method indicates if closing the file when it hasn't been saved is a reason to alert the user. See also `can-close?`.

- (send *an-editor:file* allow-close-with-no-filename?) ⇒ **boolean**

Defaultly returns #f.

`get-can-close-parent`

The result of this method is used as the parent for the dialog that asks about closing.

```
- (send an-editor:file get-can-close-parent) ⇒ (union false (is-a/c? frame%) (is-a/c? dialog%))
```

Defaultly returns #f.

update-frame-filename

Attempts to find a frame that displays this editor. If it does, it updates the frame's title based on a new filename in the editor.

```
- (send an-editor:file update-frame-filename) ⇒ void
```

8.10 editor:file-mixin

Domain: `editor:keymap<%>`

Implements: `editor:file<%>`

Implements: `editor:keymap<%>`

This editor locks itself when the file that is opened is read-only in the filesystem.

The class that this mixin produces uses the same initialization arguments as it's input.

`can-close?` (*augments, and augmentable only*)

This method is called to query the editor if is okay to close the editor. Although there is no visible effect associated with closing an editor, there may be some cleanup actions that need to be run when the user is finished with the editor (asking if it should be saved, for example).

See also `on-close` and `close`.

```
- (send an-editor:file-mixin can-close? boolean)
```

If the `allow-close-with-no-filename?` method returns #f, this method checks to see if the file has been saved at all yet. If not, it asks the user about saving (and saves if they ask).

If the `allow-close-with-no-filename?` method returns #t, this method does as before, except only asks if the editor's `get-filename` method returns a path.

Also calls `inner`.

get-keymaps

The keymaps returned from this method are chained to this `editor<%>`'s keymap.

```
- (send an-editor:file-mixin get-keymaps (list-of (instance keymap%)))
```

This returns a list containing the super-class's keymaps, plus the result of `keymap:get-file`

```
set-filename
```

Set the path name for the file to be saved from or reloaded into this editor. This method is also called when the filename changes through any method (such as `load-file`).

```
- (send an-editor:file-mixin set-filename name temp? void
  name: string
  temp? = #f: boolean
```

Updates the filename on each frame displaying this editor, for each frame that matches `frame:editor<%>`.

8.11 editor:backup-autosave<%>

Extends: `editor:basic<%>`

Classes matching this interface support backup files and autosaving.

```
autosave?
```

Indicates weather this `editor<%>` should be autosaved.

```
- (send an-editor:backup-autosave autosave?) => boolean
```

Returns #t.

```
backup?
```

Indicates weather this `editor<%>` should be backed up.

```
- (send an-editor:backup-autosave backup?) => boolean
```

Returns the value of the `preferences:get` applied to `'framework:backup-files?`.

```
do-autosave
```

This method is called to perform the autosaving. See also `autosave:register`

- (send *an-editor:backup-autosave* do-autosave) ⇒ (union #f string)

When the file has been modified since it was last saved and autosaving it turned on (via the `autosave?` method) an autosave file is created for this `editor<%>`.

Returns the filename where the autosave took place, or #f if none did.

`remove-autosave`

This method removes the autosave file associated with this `editor<%>`.

- (send *an-editor:backup-autosave* remove-autosave) ⇒ void

8.12 editor:backup-autosave-mixin

Domain: `editor:basic<%>`

Implements: `autosave:autosavable<%>`

Implements: `editor:basic<%>`

Implements: `editor:backup-autosave<%>`

This mixin adds backup and autosave functionality to an editor.

During initialization, this object is registered with `autosave:register`.

The result of this mixin uses the same initialization arguments as the mixin's argument.

`on-change` (*augments, and augmentable only*)

Called whenever any change is made to the editor that affects the way the editor is drawn or the values reported for the `location`/size of some snip in the editor. The `on-change` method is called just before the editor calls its administrator's `needs-update` method to refresh the editor's `display`, and it is also called just before and after printing an editor.

The editor is locked for writing and reflowing during the call to `on-change`.

- (send *an-editor:backup-autosave-mixin* on-change void

Sets a flag indicating that this `editor<%>` needs to be autosaved.

`on-close` (*augments, and augmentable only*)

This method is called when an editor is closed. See also `can-close?` and `close`.

- (send *an-editor:backup-autosave-mixin* on-close void
Deletes the autosave file and turns off autosaving.

on-save-file (*augments, and augmentable only*)

Called just before the editor is saved to a file, after calling `can-save-file?` to verify that the save is allowed. See also `after-save-file`.

- (send *an-editor:backup-autosave-mixin* on-save-file *filename* *format* bool
filename: path
format: symbol in '(guess standard text text-force-cr same copy)

If a backup file has not been created this session for this file, deletes any existing backup file and copies the old save file into the backup file. For the backup file's name, see `path-utils:generate-backup-name`

set-modified

Sets the modified state of the editor. Usually, the state is changed automatically after an insertion, deletion, or style change by calling this method. (This method is also called when the modification state changes through *any* method.) This method is usually not called when the state of the flag is not changing.

See also `is-modified?` and `on-snip-modified`.

- (send *an-editor:backup-autosave-mixin* set-modified *modified?* void
modified?: boolean

If the file is no longer modified, this method deletes the autosave file. If it is, it updates a flag to indicate that the autosave file is out of date.

8.13 editor:info<%>

Extends: `editor:basic<%>`

An `editor<%>` matching this interface provides information about its lock state to its `top-level-window<%>`.

8.14 editor:info-mixin

Domain: `editor:basic<%>`

Implements: `editor:info<%>`

Implements: `editor:basic<%>`

This editor tells the frame when it is locked and unlocked. See also `frame:text-info<%>`.

`lock`

Locks or unlocks the editor for modifications. If an editor is locked, *all* modifications are blocked, not just user modifications.

See also `is-locked?`.

This method does not affect internal locks, as discussed in “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*).

```
- (send an-editor:info-mixin lock lock? void
  lock? : boolean
```

Uses `run-after-edit-sequence` to call `lock-status-changed`.

9. Exit

The exiting library provides a way to perform cleanup actions when the application is quit.

Clean up actions can be installed by registering a callback procedure that will be invoked by `exit:exit`.

On exit, *callback* will be called with no arguments. Also, use `exit:insert-on-callback:`

```
(exit:insert-on-callback callback)
```

to perform cleanup actions.

Also, callbacks can be registered that abort exiting. To install such a callback, use `exit:insert-can?-callback:`

```
(exit:insert-can?-callback callback)
```

if *callback* returns #f, then the exit is aborted.

10. Exceptions

11. Finder

The procedures `finder:get-file` and `finder:put-file` query the user for a filename; `finder:get-file` gets the name of an existing file, and `finder:put-file` gets the name for a new file.

Under Windows and MacOS, `finder:get-file` and `finder:put-file` call `finder:std-get-file` and `finder:std-put-file`, which implement the filename query using platform-specific dialogs. Under Unix, `finder:get-file` and `finder:put-file` call `finder:common-get-file` and `finder:common-put-file` which use a platform-independent dialog. Which procedure `finder:get-file` and `finder:put-file` call depends on the value of the preference (see [fw:preferences](#)) `'framework:file-dialogs`. It is `'common`, the common platform-independent dialogs are used and if it is `'std`, the standard platform-specific dialogs are used.

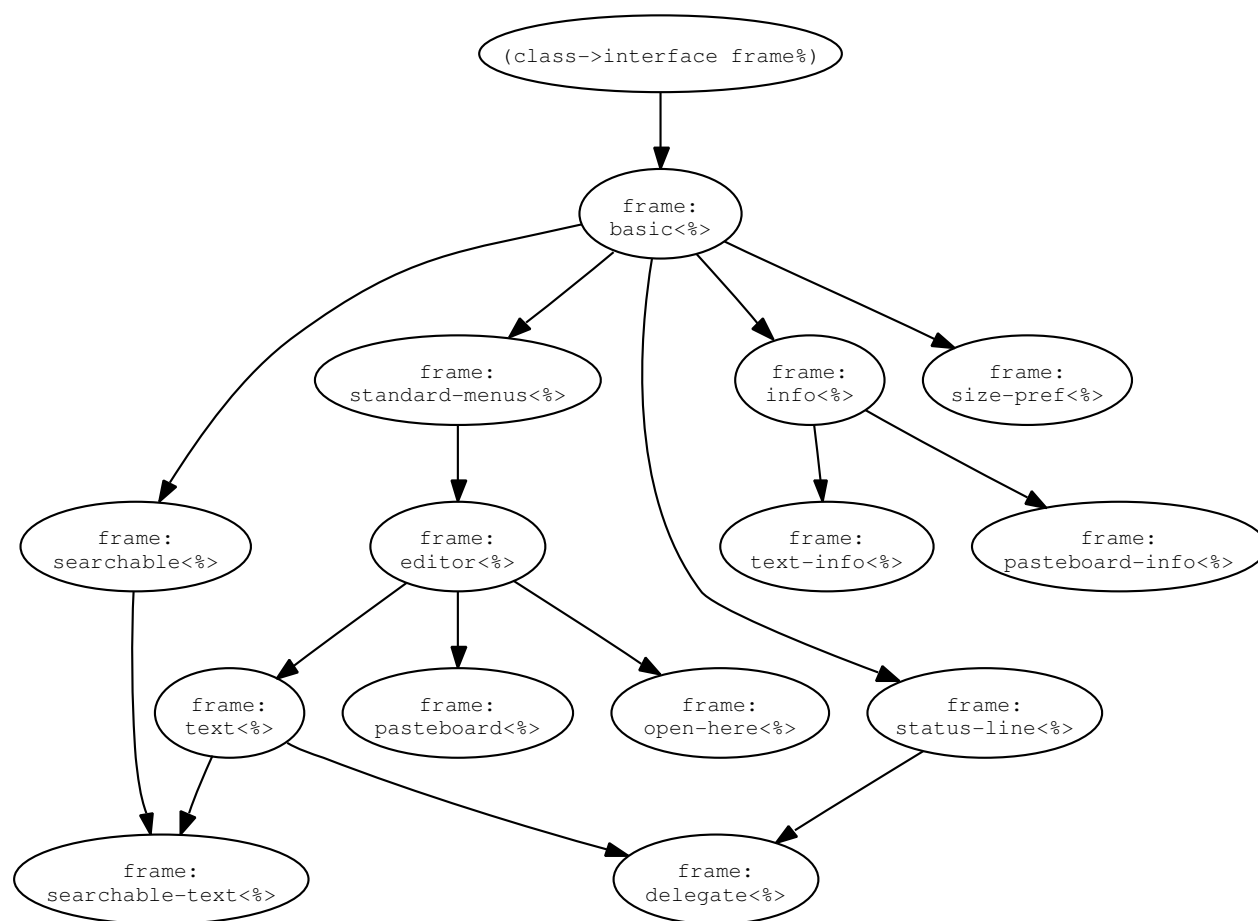
The procedure `finder:common-get-file-list` gets a list of filenames from the user using a platform-independent dialog. The arguments are the same as for `finder:get-file` and the result is either `#f` or a list of filenames.

All filenames returned by these procedures are normalized using `normalize-path` from [mz:mzlibfile](#).

12. Frame

This chapter describes the frame mixins, interfaces, and classes.

This is the interface hierarchy:



- frame:basic%
- frame:delegate%
- frame:editor%
- frame:info%
- frame:open-here%

- `frame:pasteboard-info%`
- `frame:pasteboard%`
- `frame:searchable%`
- `frame:size-pref%`
- `frame:standard-menus%`
- `frame:status-line%`
- `frame:text-info%`
- `frame:text%`
- `frame:basic<%>`
- `frame:delegate<%>`
- `frame:editor<%>`
- `frame:info<%>`
- `frame:open-here<%>`
- `frame:pasteboard-info<%>`
- `frame:pasteboard<%>`
- `frame:register-group<%>`
- `frame:searchable-text<%>`
- `frame:searchable<%>`
- `frame:size-pref<%>`
- `frame:standard-menus<%>`
- `frame:status-line<%>`
- `frame:text-info<%>`
- `frame:text<%>`
- `frame:basic-mixin`
- `frame:delegate-mixin`
- `frame:editor-mixin`
- `frame:info-mixin`
- `frame:open-here-mixin`
- `frame:pasteboard-info-mixin`
- `frame:pasteboard-mixin`
- `frame:register-group-mixin`
- `frame:searchable-mixin`
- `frame:searchable-text-mixin`

- frame:size-pref-mixin
- frame:standard-menus-mixin
- frame:status-line-mixin
- frame:text-info-mixin
- frame:text-mixin

12.1 frame:basic<%>

Extends: (class->interface **frame%**)

Classes matching this interface support the basic **frame%** functionality required by the framework.

close

This method closes the frame by calling the **can-close?**, **on-close**, and **show** methods.

It's implementation is:

```
(inherit can-close? on-close)
(public
 [show
  (lambda ()
    (when (can-close?)
      (on-close)
      (show #f)))]])
```

- (send *a-frame:basic* close) ⇒ void

editing-this-file?

Indicates if this frame contains this buffer (and can edit that file).

```
- (send a-frame:basic editing-this-file? filename) ⇒ boolean
  filename: path
```

Returns #f.

get-area-container

This returns the main **area-container<%>** in the frame

```
- (send a-frame:basic get-area-container) ⇒ (instance (implements area-container<%>))
```

get-area-container%

The class that this method returns is used to create the `area-container<%>` in this frame.

```
- (send a-frame:basic get-area-container%) => (implements area-container<%>)
```

get-filename

This returns the filename that the frame is currently being saved as, or #f if there is no appropriate filename.

```
- (send a-frame:basic get-filename temp) => (union #f string)
  temp = #f : (union #f (box boolean))
```

Defaultly returns #f.

If *temp* is a box, it is filled with #t or #f, depending if the filename is a temporary filename.

get-menu-bar%

The result of this method is used to create the initial menu bar for this frame.

```
- (send a-frame:basic get-menu-bar%) => (derived-from menu-bar%)
```

Return `menu-bar%`.

make-root-area-container

Override this method to insert a panel in between the panel used by the clients of this frame and the frame itself. For example, to insert a status line panel override this method with something like this:

```
(class ...
  ...
  (rename [super-make-root-area-container make-root-area-container])
  (field
    [status-panel #f])
  (define/override (make-root-area-container cls parent)
    (set! status-panel
          (super-make-root-area-container vertical-panel% parent))
    (let ([root (make-object cls status-panel)]

          ; ... add other children to status-panel ...

      root))
  ...
```

In this example, `status-panel` will contain a root panel for the other classes, and whatever panels are needed to display status information.

The searching frame is implemented using this method.

```
- (send a-frame:basic make-root-area-container class parent) => (instance (implements area-container<%>))
  class: (implements area-container<%>)
  parent: (instance (implements area-container<%>))
  Calls make-object with class and parent.
```

make-visible

Makes the file named by *filename* visible (intended for use with tabbed editing).

```
- (send a-frame:basic make-visible filename) => void
  filename: string
```

12.2 frame:basic-mixin

Domain: (class->interface **frame%**)

Implements: **frame:basic<%>**

This mixin provides the basic functionality that the framework expects. It helps manage the list of frames in the **group:%** object returned by **group:get-the-frame-group**.

Do not give **panel%**s or **control<%>**s this frame as parent. Instead, use the result of the **get-area-container** method.

This mixin also creates a menu bar for the frame, as the frame is initialized. It uses the class returned by **get-menu-bar%**. It only passes the frame as an initialization argument. In addition, it creates the windows menu in the menu bar.

See also **frame:reorder-menus**.

```
- init args: (label -) [(parent -)] [(width -)] [(height -)] [(x -)] [(y -)] [(style -)] [(enabled -)] [(border -)] [(spacing -)] [(alignment -)] [(min-width -)] [(min-height -)] [(stretchable-width -)] [(stretchable-height -)]
  label: string (up to 200 characters)
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu mdi-parent mdi-child toolbar-button hide-menu-bar float metal)
  enabled = #t: boolean
```

```

border = 0 : exact integer in [0, 1000]
spacing = 0 : exact integer in [0, 1000]
alignment = '(center top) : two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
min-width = 0 : exact integer in [0, 10000]
min-height = 0 : exact integer in [0, 10000]
stretchable-width = #t : boolean
stretchable-height = #t : boolean

```

The *label* string is displayed in the frame's title bar. If the frame's label is changed (see `set-label`), the title bar is updated.

The *parent* argument can be `#f` or an existing frame. Under Windows, if *parent* is an existing frame, the new frame is always on top of its parent. Also, the *parent* frame may be an MDI parent frame from a new MDI child frame. Under Windows and X (for many window managers), a frame is iconized when its parent is iconized.

If *parent* is `#f`, then the eventspace for the new frame is the current eventspace, as determined by `current-eventspace`. Otherwise, *parent*'s eventspace is the new frame's eventspace.

If the *width* or *height* argument is not `#f`, it specifies an initial size for the frame (in pixels) assuming that it is larger than the minimum size, otherwise the minimum size is used.

If the *x* or *y* argument is not `#f`, it specifies an initial location for the frame. Otherwise, a location is selected automatically (tiling frames and dialogs as they are created).

The *style* flags adjust the appearance of the frame on some platforms:

- `'no-resize-border` — omits the resizable border around the window (Windows, X MWM) or grow box in the bottom right corner (Mac OS X)
- `'no-caption` — omits the title bar for the frame (Windows, X MWM) (X Gnome, X KDE: the frame decoration is omitted completely when `'no-resize-border` and `'no-caption` are combined.)
- `'no-system-menu` — omits the system menu (Windows)
- `'mdi-child` — creates the frame as a MDI (multiple document interface) child frame, mutually exclusive with `'mdi-parent` (Windows)
- `'mdi-parent` — creates the frame as a MDI (multiple document interface) parent frame, mutually exclusive with `'mdi-child` (Windows)
- `'toolbar-button` — includes a toolbar button on the frame's title bar (Mac OS X); a click on the toolbar button triggers a call to `on-toolbar-button-click`
- `'hide-menu-bar` — hides the menu bar and dock when the frame is active (Mac OS X)
- `'float` — causes the frame to stay in front of all other non-floating windows (Windows and Mac OS X always, X when combined with `'no-caption`); under Mac OS X, a floating frame shares the focus with an active non-floating frame; when this style is combined with `'no-caption`, then showing the frame does not cause the keyboard focus to shift to the window, and under X, clicking the frame does not move the focus
- `'metal` — draws the frame with a brushed-metal background (Mac OS X); this style is ignored when `'no-caption` is specified

If the `'mdi-child` style is specified, the *parent* must be a frame with the `'mdi-parent` style, otherwise an `exn:fail:contract` exception is raised.

Even if the frame is not shown, a few notification events may be queued for the frame on creation. Consequently, the new frame's resources (e.g., memory) cannot be reclaimed until some events are handled, or the frame's eventspace is shut down.

For information about the *enabled* argument, see `window<%>`. For information about the *border*, *spacing*, and *alignment* arguments, see `area-container<%>`. For information about the *min-width*, *min-height*, *stretchable-width*, and *stretchable-height* arguments, see `area<%>`.

after-new-child

This method is called after a new containee area is created with this area as its container. The new child is provided as an argument to the method.

- (send *a-frame:basic-mixin* after-new-child void
 Raises an exception if attempting to add a child to this frame (except if using the `make-root-area-container` method).

can-exit?

Called before `on-exit` to check whether an exit is allowed. See `on-exit` for more information.

- (send *a-frame:basic-mixin* can-exit? boolean
 This, together with `on-exit` mimics `exit:exit`.
 First, it calls `exit:set-exiting` with #t. Then, it calls `exit:can-exit?`. If it returns #t, so does this method. If it returns #f, this method calls `exit:set-exiting` with #f.

on-drop-file

For platforms that support drag-and-drop, this method is called when the user drags a file onto the window. Drag-and-drop must first be enabled for the window with `accept-drop-files`.

Under Mac OS X, when the application is running and user double-clicks an application-handled file or drags a file onto the application's icon, the main thread's application file handler is called (see `application-file-handler`). The default handler calls the `on-drop-file` method of the most-recently activated frame if drag-and-drop is enabled for that frame, independent of the frame's eventspace (but the method is called in the frame's eventspace's handler thread). When the application is not running, the filenames will be provided as command-line arguments.

- (send *a-frame:basic-mixin* on-drop-file *pathname* void
pathname: string
 Calls `handler:edit-file` with *pathname* as an argument.

on-exit

Called by the default application quit handler (as determined by the `application-quit-handler` parameter) when the operating system requests that the application shut down (e.g., when the Quit menu item is selected in the main application menu under Mac OS X). In that case, this method is called for the most recently active top-level window in the initial eventspace, but only if the window's `can-exit?` method first returns true.

- (send *a-frame:basic-mixin* on-exit void
 Together with `can-exit?` this mimics the behavior of `exit:exit`.

Calls `exit:on-exit` and then queues a callback to call MzScheme's `exit` function. If that returns, it calls `exit:set-exiting` to reset that flag to `#f`.

`on-superwindow-show`

Called via the event queue whenever the visibility of a window has changed, either through a call to the window's `show`, through the showing/hiding of one of the window's ancestors, or through the activating or deactivating of the window or its ancestor in a container (e.g., via `delete-child`). The method's argument indicates whether the window is now visible or not.

This method is not called when the window is initially created; it is called only after a change from the window's initial visibility. Furthermore, if a show notification event is queued for the window and it reverts its visibility before the event is dispatched, then the dispatch is canceled.

```
- (send a-frame:basic-mixin on-superwindow-show shown? void
    shown?: boolean
```

Notifies the result of (`group:get-the-frame-group`) that a frame has been shown, by calling the `frame-shown/hidden` method.

`show`

Shows or hides a window.

The visibility of a window can be changed by the user clicking the window's close box, for example, and such changes do not go through this method; use `on-superwindow-show` or `on-close` to monitor visibility changes.

```
- (send a-frame:basic-mixin show on? void
    on?: boolean
```

Calls the super method.

When `on?` is `#t`, inserts the frame into the frame group and when it is `#f`, removes the frame from the group.

12.3 frame:size-pref<%>

Extends: `frame:basic<%>`

12.4 frame:size-pref-mixin

Domain: `frame:basic<%>`

Implements: `frame:basic<%>`

Implements: `frame:size-pref<%>`

```
- init args:  size-preferences-key
             size-preferences-key: symbol
```

The size `size-preferences-key` symbol is used with `preferences:get` and `preferences:get` to track the current size.

Passes the `width` and `height` initialization arguments to the superclass based on the current value of the preference.

See also `frame:setup-size-pref`.

`on-size`

Called when the window is resized. The window's new size (in pixels) is provided to the method. The size values are for the entire window, not just the client area.

```
- (send a-frame:size-pref-mixin on-size width height void
    width: number
    height: number
```

Updates the preferences, according to the width and height. The preferences key is the one passed to the initialization argument of the class.

12.5 `frame:register-group<%>`

Frames that implement this interface are registered with the group. See `group:get-the-frame-group` and `frame:register-group-mixin`.

12.6 `frame:register-group-mixin`

Domain: `frame:basic<%>`

Implements: `frame:register-group<%>`

Implements: `frame:basic<%>`

During initialization, calls `insert-frame` with `this`.

`can-close?` (*augments, and augmentable only*)

Called just before the window might be closed (e.g., by the window manager). If #f is returned, the window is not closed, otherwise `on-close` is called and the window is closed (i.e., the window is hidden, like calling `show` with #f).

This method is *not* called by `show`.

```
- (send a-frame:register-group-mixin can-close? bool
```

Calls the inner method, with a default of #t. If that returns #t, it checks for one of the these three conditions:

- `exit:exiting?` returns #t
- there is more than one frame in the group returned by `group:get-the-frame-group`, or
- the procedure `exit:user-oks-exit` returns #t.

If any of those conditions hold, the method returns #t.

`on-activate`

Called when a window is *activated* or *deactivated*. A top-level window is activated when the keyboard focus moves from outside the window to the window or one of its children. It is deactivated when the focus moves back out of the window. Under Mac OS X, a child of a floating frames can have the focus instead of a child of the active non-floating frame; in other words, floating frames act as an extension of the active non-frame for keyboard focus.

The method's argument is #t when the window is activated, #f when it is deactivated.

```
- (send a-frame:register-group-mixin on-activate on? void
  on?: boolean
```

Calls `set-active-frame` with `this` when `on?` is true.

`on-close` (*augments, and augmentable only*)

Called just before the window is closed (e.g., by the window manager). This method is *not* called by `show`.

See also `can-close?`.

```
- (send a-frame:register-group-mixin on-close void
```

First calls the inner method. Next, calls the `remove-frame` method of the result of `group:get-the-frame-group` with `this` as an argument. Finally, unless `exit:exiting?` returns #t, and if there are no more frames open, it calls `exit:exit`.

12.7 frame:status-line<%>

Extends: `frame:basic<%>`

The mixin that implements this interface provides an interface to a set of status lines at the bottom of this frame.

Each status line must be opened with `open-status-line` before any messages are shown in the status line and once `close-status-line` is called, no more messages may be displayed, unless the status line is re-opened.

The screen space for status lines is not created until `update-status-line` is called with a string. Additionally, the screen space for one status line is re-used when by another status line when the first passes `#f` to `update-status-line`. In this manner, the status line frame avoids opening too many status lines and avoids flashing the status lines open and closed too often.

`close-status-line`

Closes the status line *id*.

```
- (send a-frame:status-line close-status-line id) ⇒ void
  id: symbol
```

`open-status-line`

Creates a new status line identified by the symbol argument. The line will not appear in the frame until a message is put into it, via `update-status-line`.

```
- (send a-frame:status-line open-status-line id) ⇒ void
  id: symbol
```

`update-status-line`

Updates the status line named by *id* with *status*. If *status* is `#f`, the status line is becomes blank (and may be used by other ids).

```
- (send a-frame:status-line update-status-line id status) ⇒ void
  id: symbol
  status: (union #f string)
```

12.8 frame:status-line-mixin

Domain: `frame:basic<%>`

Implements: `frame:basic<%>`

Implements: `frame:status-line<%>`

make-root-area-container

Override this method to insert a panel in between the panel used by the clients of this frame and the frame itself. For example, to insert a status line panel override this method with something like this:

```
(class ...
  ...
  (rename [super-make-root-area-container make-root-area-container])
  (field
    [status-panel #f])
  (define/override (make-root-area-container cls parent)
    (set! status-panel
      (super-make-root-area-container vertical-panel% parent))
    (let ([root (make-object cls status-panel)]))

      ; ... add other children to status-panel ...

      root))
  ...)
```

In this example, status-panel will contain a root panel for the other classes, and whatever panels are needed to display status information.

The searching frame is implemented using this method.

```
- (send a-frame:status-line-mixin make-root-area-container class parent (im-
  plements panel%)
  class: (derived-from panel%)
  parent: (instanceof (derived-from panel%)))
```

Adds a panel at the bottom of the frame to hold the status lines.

12.9 frame:info<%>

Extends: `frame:basic<%>`

Frames matching this interface support a status line.

The preference `'framework:show-status-line` controls the visibility of the status line. If it is `#t`, the status line is visible and if it is `#f`, the status line is not visible (see [fw:preferences](#)).

determine-width

This method is used to calculate the size of an `editor-canvas%` with a particular set of characters in it. It is used to calculate the sizes of the edits in the status line.

```
- (send a-frame:info determine-width str canvas text) ⇒ integer
  str: string
  canvas: (instance editor-canvas%)
  text: (instance text%)
```

get-info-canvas

Returns the canvas that the `frame:info<%>` currently shows info about. See also `set-info-canvas`

```
- (send a-frame:info get-info-canvas) ⇒ (instance canvas:basic%)
```

get-info-editor

Override this method to specify the editor that the status line contains information about.

```
- (send a-frame:info get-info-editor) ⇒ (union #f (implements editor<%>))
  Returns the result of get-editor.
```

get-info-panel

This method returns the panel where the information about this editor is displayed.

```
- (send a-frame:info get-info-panel) ⇒ (instance horizontal-panel%)
```

hide-info

Hides the info panel.

See also `is-info-hidden?`.

```
- (send a-frame:info hide-info) ⇒ void
```

is-info-hidden?

Result indicates if the show info panel has been explicitly hidden with `hide-info`.

If this method returns `#t` and `(preferences:get 'framework:show-status-line)` is `#f`, then the info panel will not be visible. Otherwise, it is visible.

```
- (send a-frame:info is-info-hidden?) ⇒ boolean
```

lock-status-changed

This method is called when the lock status of the `editor<%>` changes.

- (send `a-frame:info` lock-status-changed) ⇒ void
Updates the lock icon in the status line panel.

set-info-canvas

Sets this canvas to be the canvas that the info frame shows info about. The `on-focus` and `set-editor` methods call this method to ensure that the info canvas is set correctly.

- (send `a-frame:info` set-info-canvas `canvas`) ⇒ void
`canvas`: (instance `canvas:basic%`)

show-info

Shows the info panel.

See also `is-info-hidden?`.

- (send `a-frame:info` show-info) ⇒ void

update-info

This method updates all of the information in the panel.

- (send `a-frame:info` update-info) ⇒ void

12.10 frame:info-mixin

Domain: `frame:basic<%>`

Implements: `frame:info<%>`

Implements: `frame:basic<%>`

This mixin provides support for displaying various info in the status line of the frame.

The result of this mixin uses the same initialization arguments as the mixin's argument.

make-root-area-container

Override this method to insert a panel in between the panel used by the clients of this frame and the frame itself. For example, to insert a status line panel override this method with something like this:

```
(class ...
  ...
  (rename [super-make-root-area-container make-root-area-container])
  (field
    [status-panel #f])
  (define/override (make-root-area-container cls parent)
    (set! status-panel
          (super-make-root-area-container vertical-panel% parent))
    (let ([root (make-object cls status-panel)])

      ; ... add other children to status-panel ...

      root))
  ...)
```

In this example, *status-panel* will contain a root panel for the other classes, and whatever panels are needed to display status information.

The searching frame is implemented using this method.

```
- (send a-frame:info-mixin make-root-area-container class parent (instance area-container<%>
  class: (implements area-container<%>))
  parent: (instance (implements area-container<%>)))
```

Builds an extra panel for displaying various information.

on-close (*augments, and augmentable only*)

Called just before the window is closed (e.g., by the window manager). This method is *not* called by *show* .

See also *can-close?*.

```
- (send a-frame:info-mixin on-close void)
```

Removes the GC icon with *unregister-collecting-blit* and cleans up other callbacks.

12.11 frame:text-info<%>

Extends: *frame:info*<%>

Objects matching this interface receive information from editors constructed with `editor:info-mixin` and display it.

`anchor-status-changed`

This method is called when the anchor is turned either on or off in the `editor<%>` in this frame.

```
- (send a-frame:text-info anchor-status-changed) ⇒ void
```

`editor-position-changed`

This method is called when the position in the `editor<%>` changes.

```
- (send a-frame:text-info editor-position-changed) ⇒ void
```

`overwrite-status-changed`

This method is called when the overwrite mode is turned either on or off in the `editor<%>` in this frame.

```
- (send a-frame:text-info overwrite-status-changed) ⇒ void
```

`set-macro-recording`

Shows/hides the icon in the info bar that indicates if a macro recording is in progress.

```
- (send a-frame:text-info set-macro-recording on?) ⇒ void
  on?: boolean
```

12.12 frame:text-info-mixin

Domain: `frame:info<%>`

Implements: `frame:info<%>`

Implements: `frame:text-info<%>`

This mixin adds status information to the info panel relating to an edit.

`on-close` (*augments, and augmentable only*)

Called just before the window is closed (e.g., by the window manager). This method is *not* called by `show`.

See also `can-close?`.

- (send `a-frame:text-info-mixin` `on-close` `void`
removes a preferences callback for `'framework:line-offsets`. See section 23 for more information

`update-info`

This method updates all of the information in the panel.

- (send `a-frame:text-info-mixin` `update-info` `void`
Calls `overwrite-status-changed`, `anchor-status-changed`, and `editor-position-changed`.

12.13 `frame:pasteboard-info<%>`

Extends: `frame:info<%>`

12.14 `frame:pasteboard-info-mixin`

Domain: `frame:basic<%>`

Implements: `frame:pasteboard-info<%>`

Implements: `frame:basic<%>`

12.15 `frame:standard-menus<%>`

Extends: `frame:basic<%>`

- `file-menu:new-callback`, `file-menu:create-new?`, `file-menu:new-string`, `file-menu:new-help-st`
`file-menu:new-on-demand`, `file-menu:get-new-item`

- `file-menu:between-new-and-open`
- `file-menu:open-callback`, `file-menu:create-open?`, `file-menu:open-string`, `file-menu:open-help-string`, `file-menu:open-on-demand`, `file-menu:get-open-item`
- `file-menu:open-recent-callback`, `file-menu:create-open-recent?`, `file-menu:open-recent-string`, `file-menu:open-recent-help-string`, `file-menu:open-recent-on-demand`, `file-menu:get-open-recent-item`
- `file-menu:between-open-and-revert`
- `file-menu:revert-callback`, `file-menu:create-revert?`, `file-menu:revert-string`, `file-menu:revert-help-string`, `file-menu:revert-on-demand`, `file-menu:get-revert-item`
- `file-menu:between-revert-and-save`
- `file-menu:save-callback`, `file-menu:create-save?`, `file-menu:save-string`, `file-menu:save-help-string`, `file-menu:save-on-demand`, `file-menu:get-save-item`
- `file-menu:save-as-callback`, `file-menu:create-save-as?`, `file-menu:save-as-string`, `file-menu:save-as-help-string`, `file-menu:save-as-on-demand`, `file-menu:get-save-as-item`
- `file-menu:between-save-as-and-print`
- `file-menu:print-callback`, `file-menu:create-print?`, `file-menu:print-string`, `file-menu:print-help-string`, `file-menu:print-on-demand`, `file-menu:get-print-item`
- `file-menu:between-print-and-close`
- `file-menu:close-callback`, `file-menu:create-close?`, `file-menu:close-string`, `file-menu:close-help-string`, `file-menu:close-on-demand`, `file-menu:get-close-item`
- `file-menu:between-close-and-quit`
- `file-menu:quit-callback`, `file-menu:create-quit?`, `file-menu:quit-string`, `file-menu:quit-help-string`, `file-menu:quit-on-demand`, `file-menu:get-quit-item`
- `file-menu:after-quit`
- `edit-menu:undo-callback`, `edit-menu:create-undo?`, `edit-menu:undo-string`, `edit-menu:undo-help-string`, `edit-menu:undo-on-demand`, `edit-menu:get-undo-item`
- `edit-menu:redo-callback`, `edit-menu:create-redo?`, `edit-menu:redo-string`, `edit-menu:redo-help-string`, `edit-menu:redo-on-demand`, `edit-menu:get-redo-item`
- `edit-menu:between-redo-and-cut`
- `edit-menu:cut-callback`, `edit-menu:create-cut?`, `edit-menu:cut-string`, `edit-menu:cut-help-string`, `edit-menu:cut-on-demand`, `edit-menu:get-cut-item`
- `edit-menu:between-cut-and-copy`
- `edit-menu:copy-callback`, `edit-menu:create-copy?`, `edit-menu:copy-string`, `edit-menu:copy-help-string`, `edit-menu:copy-on-demand`, `edit-menu:get-copy-item`
- `edit-menu:between-copy-and-paste`
- `edit-menu:paste-callback`, `edit-menu:create-paste?`, `edit-menu:paste-string`, `edit-menu:paste-help-string`, `edit-menu:paste-on-demand`, `edit-menu:get-paste-item`
- `edit-menu:between-paste-and-clear`

- `edit-menu:clear-callback`, `edit-menu:create-clear?`, `edit-menu:clear-string`, `edit-menu:clear`, `edit-menu:clear-on-demand`, `edit-menu:get-clear-item`
- `edit-menu:between-clear-and-select-all`
- `edit-menu:select-all-callback`, `edit-menu:create-select-all?`, `edit-menu:select-all-string`, `edit-menu:select-all-help-string`, `edit-menu:select-all-on-demand`, `edit-menu:get-select-all`
- `edit-menu:between-select-all-and-find`
- `edit-menu:find-callback`, `edit-menu:create-find?`, `edit-menu:find-string`, `edit-menu:find-help-string`, `edit-menu:find-on-demand`, `edit-menu:get-find-item`
- `edit-menu:find-again-callback`, `edit-menu:create-find-again?`, `edit-menu:find-again-string`, `edit-menu:find-again-help-string`, `edit-menu:find-again-on-demand`, `edit-menu:get-find-again`
- `edit-menu:replace-and-find-again-callback`, `edit-menu:create-replace-and-find-again?`, `edit-menu:replace-and-find-again-string`, `edit-menu:replace-and-find-again-help-string`, `edit-menu:replace-and-find-again-on-demand`, `edit-menu:get-replace-and-find-again-item`
- `edit-menu:between-find-and-preferences`
- `edit-menu:preferences-callback`, `edit-menu:create-preferences?`, `edit-menu:preferences-string`, `edit-menu:preferences-help-string`, `edit-menu:preferences-on-demand`, `edit-menu:get-preferences`
- `edit-menu:after-preferences`
- `help-menu:before-about`
- `help-menu:about-callback`, `help-menu:create-about?`, `help-menu:about-string`, `help-menu:about-help-string`, `help-menu:about-on-demand`, `help-menu:get-about-item`
- `help-menu:after-about`

`edit-menu:after-preferences`

This method is called after the addition of the preferences menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus edit-menu:after-preferences menu) => void
  menu: (instance (derived-from menu%))
```

Does nothing.

`edit-menu:between-clear-and-select-all`

This method is called between the addition of the clear menu-item and before the addition of the select-all menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus edit-menu:between-clear-and-select-all menu) =>
void
  menu: (instance (derived-from menu%))
```

Does nothing.

edit-menu:between-copy-and-paste

This method is called between the addition of the copy menu-item and before the addition of the paste menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus edit-menu:between-copy-and-paste menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

edit-menu:between-cut-and-copy

This method is called between the addition of the cut menu-item and before the addition of the copy menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus edit-menu:between-cut-and-copy menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

edit-menu:between-find-and-preferences

This method is called between the addition of the find menu-item and before the addition of the preferences menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus edit-menu:between-find-and-preferences menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Adds a separator except when current-eventspace-has-standard-menus? returns #t.

edit-menu:between-paste-and-clear

This method is called between the addition of the paste menu-item and before the addition of the clear menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus edit-menu:between-paste-and-clear menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

edit-menu:between-redo-and-cut

This method is called between the addition of the redo menu-item and before the addition of the cut menu-item to the edit-menu menu. Override it to add additional menus at that point.

- (send `a-frame:standard-menus` `edit-menu:between-redo-and-cut` `menu`) ⇒ `void`
`menu: (instance (derived-from menu%))`
 Adds a separator menu item.

`edit-menu:between-select-all-and-find`

This method is called between the addition of the select-all menu-item and before the addition of the find menu-item to the edit-menu menu. Override it to add additional menus at that point.

- (send `a-frame:standard-menus` `edit-menu:between-select-all-and-find` `menu`) ⇒ `void`
`menu: (instance (derived-from menu%))`
 Adds a separator menu item.

`edit-menu:clear-callback`

This method is called when the clear menu-item of the edit-menu menu is selected.

- (send `a-frame:standard-menus` `edit-menu:clear-callback` `item` `evt`) ⇒ `void`
`item: (instance (derived-from menu-item%))`
`evt: (instance control-event%)`
 Defaultly bound to:

```
(
  (menu evt)
  (let ((edit (get-edit-target-object)))
    (when (and edit (is-a? edit editor<%>))
      (send edit do-edit-operation 'clear)))
  #t)
```

`edit-menu:clear-help-string`

This result of this method is used as the help string when the `menu-item%` object is created.

- (send `a-frame:standard-menus` `edit-menu:clear-help-string`) ⇒ `string`
 Defaultly returns `"(string-constant clear-info)"`

`edit-menu:clear-on-demand`

The menu item's on-demand method calls this method

- (send `a-frame:standard-menus` `edit-menu:clear-on-demand` `item`) ⇒ `void`
`item: menu-item%`

Defaultly is this: ((item) (let* ((editor (get-edit-target-object)) (enable? (and editor (is-a? editor editor<\%>) (send editor can-do-edit-operation? (quote clear)))))) (send item enable enable?)))

edit-menu:clear-string

The result of this method is the name of this menu.

- (send a-frame:standard-menus edit-menu:clear-string) ⇒ string
defaultly returns "(if (eq? (system-type) (quote windows)) (string-constant clear-menu-item-windows) (string-constant clear-menu-item-windows))"

edit-menu:copy-callback

This method is called when the copy menu-item of the edit-menu menu is selected.

- (send a-frame:standard-menus edit-menu:copy-callback item evt) ⇒ void
item: (instance (derived-from menu-item%))
evt: (instance control-event%)

Defaultly bound to:

```
(
(menu evt)
(let ((edit (get-edit-target-object)))
  (when (and edit (is-a? edit editor<%>))
    (send edit do-edit-operation 'copy)))
#t)
```

edit-menu:copy-help-string

This result of this method is used as the help string when the menu-item% object is created.

- (send a-frame:standard-menus edit-menu:copy-help-string) ⇒ string
Defaultly returns "(string-constant copy-info)"

edit-menu:copy-on-demand

The menu item's on-demand method calls this method

- (send a-frame:standard-menus edit-menu:copy-on-demand item) ⇒ void
item: menu-item%

Defaultly is this: `((item) (let* ((editor (get-edit-target-object)) (enable? (and editor (is-a? editor editor<\%>) (send editor can-do-edit-operation? (quote copy)))))) (send item enable enable?)))`

`edit-menu:copy-string`

The result of this method is the name of this menu.

- `(send a-frame:standard-menus edit-menu:copy-string) ⇒ string`
defaultly returns `"(string-constant copy-menu-item)"`

`edit-menu:create-clear?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- `(send a-frame:standard-menus edit-menu:create-clear?) ⇒ boolean`
defaultly returns `#t`

`edit-menu:create-copy?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- `(send a-frame:standard-menus edit-menu:create-copy?) ⇒ boolean`
defaultly returns `#t`

`edit-menu:create-cut?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- `(send a-frame:standard-menus edit-menu:create-cut?) ⇒ boolean`
defaultly returns `#t`

`edit-menu:create-find-again?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* edit-menu:create-find-again?) ⇒ **boolean**
defaultly returns #f

edit-menu:create-find?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* edit-menu:create-find?) ⇒ **boolean**
defaultly returns #f

edit-menu:create-paste?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* edit-menu:create-paste?) ⇒ **boolean**
defaultly returns #t

edit-menu:create-preferences?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* edit-menu:create-preferences?) ⇒ **boolean**
defaultly returns (not (current-eventspace-has-standard-menus?))

edit-menu:create-redo?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* edit-menu:create-redo?) ⇒ **boolean**
defaultly returns #t

edit-menu:create-replace-and-find-again?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus edit-menu:create-replace-and-find-again?) =>
  boolean
  defaultly returns #f
```

```
edit-menu:create-select-all?
```

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus edit-menu:create-select-all?) => boolean
  defaultly returns #f
```

```
edit-menu:create-undo?
```

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus edit-menu:create-undo?) => boolean
  defaultly returns #f
```

```
edit-menu:cut-callback
```

This method is called when the cut menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:cut-callback item evt) => void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
  Defaultly bound to:
  (
    (menu evt)
    (let ((edit (get-edit-target-object)))
      (when (and edit (is-a? edit editor<%>)) (send edit do-edit-operation 'cut)))
    #t)
```

```
edit-menu:cut-help-string
```

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus edit-menu:cut-help-string) => string
  Defaultly returns "(string-constant cut-info)"
```

edit-menu:cut-on-demand

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:cut-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (item) (let* ((editor (get-edit-target-object)) (enable? (and
    editor (is-a? editor editor<\%>) (send editor can-do-edit-operation? (quote
    cut)))))) (send item enable enable?)))
```

edit-menu:cut-string

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus edit-menu:cut-string) ⇒ string
  defaultly returns "(string-constant cut-menu-item)"
```

edit-menu:find-again-callback

This method is called when the find-again menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:find-again-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
  Defaultly bound to:
  ( (item control) (void))
```

edit-menu:find-again-help-string

This result of this method is used as the help string when the menu-item% object is created.

```
- (send a-frame:standard-menus edit-menu:find-again-help-string) ⇒ string
  Defaultly returns "(string-constant find-again-info)"
```

edit-menu:find-again-on-demand

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:find-again-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (item) (send item enable (let ((target (get-edit-target-object)))
    (and target (is-a? target editor<\%>))))))
```

edit-menu:find-again-string

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus edit-menu:find-again-string) ⇒ string
  defaultly returns "(string-constant find-again-menu-item)"
```

edit-menu:find-callback

This method is called when the find menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:find-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
  Defaultly bound to:
  ( (item control) (void))
```

edit-menu:find-help-string

This result of this method is used as the help string when the menu-item% object is created.

```
- (send a-frame:standard-menus edit-menu:find-help-string) ⇒ string
  Defaultly returns "(string-constant find-info)"
```

edit-menu:find-on-demand

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:find-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (item) (send item enable (let ((target (get-edit-target-object)))
    (and target (is-a? target editor<\%>))))))
```

```
edit-menu:find-string
```

The result of this method is the name of this menu.

- (send *a-frame:standard-menus* edit-menu:find-string) ⇒ string
 defaultly returns "(string-constant find-menu-item)"

```
edit-menu:get-clear-item
```

This method returns the `menu-item%` that corresponds to this menu item.

- (send *a-frame:standard-menus* edit-menu:get-clear-item) ⇒ (instance `menu-item%`)

```
edit-menu:get-copy-item
```

This method returns the `menu-item%` that corresponds to this menu item.

- (send *a-frame:standard-menus* edit-menu:get-copy-item) ⇒ (instance `menu-item%`)

```
edit-menu:get-cut-item
```

This method returns the `menu-item%` that corresponds to this menu item.

- (send *a-frame:standard-menus* edit-menu:get-cut-item) ⇒ (instance `menu-item%`)

```
edit-menu:get-find-again-item
```

This method returns the `menu-item%` that corresponds to this menu item.

- (send *a-frame:standard-menus* edit-menu:get-find-again-item) ⇒ (instance `menu-item%`)

```
edit-menu:get-find-item
```

This method returns the `menu-item%` that corresponds to this menu item.

- (send *a-frame:standard-menus* edit-menu:get-find-item) ⇒ (instance `menu-item%`)

`edit-menu:get-paste-item`

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus edit-menu:get-paste-item) => (instance menu-item%)
```

`edit-menu:get-preferences-item`

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus edit-menu:get-preferences-item) => (instance menu-item%)
```

`edit-menu:get-redo-item`

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus edit-menu:get-redo-item) => (instance menu-item%)
```

`edit-menu:get-replace-and-find-again-item`

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus edit-menu:get-replace-and-find-again-item) => (instance menu-item%)
```

`edit-menu:get-select-all-item`

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus edit-menu:get-select-all-item) => (instance menu-item%)
```

`edit-menu:get-undo-item`

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus edit-menu:get-undo-item) => (instance menu-item%)
```

edit-menu:paste-callback

This method is called when the paste menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:paste-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
(
  (menu evt)
  (let ((edit (get-edit-target-object)))
    (when (and edit (is-a? edit editor<%>))
      (send edit do-edit-operation 'paste)))
  #t)
```

edit-menu:paste-help-string

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus edit-menu:paste-help-string) ⇒ string
  Defaultly returns "(string-constant paste-info)"
```

edit-menu:paste-on-demand

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:paste-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (item) (let* ((editor (get-edit-target-object)) (enable? (and
  editor (is-a? editor editor<\%>) (send editor can-do-edit-operation? (quote
  paste)))))) (send item enable enable?)))
```

edit-menu:paste-string

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus edit-menu:paste-string) ⇒ string
  defaultly returns "(string-constant paste-menu-item)"
```

edit-menu:preferences-callback

This method is called when the preferences menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:preferences-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
( (item control) (preferences:show-dialog) #t)
```

`edit-menu:preferences-help-string`

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus edit-menu:preferences-help-string) ⇒ string
  Defaultly returns "(string-constant preferences-info)"
```

`edit-menu:preferences-on-demand`

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:preferences-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (menu-item) (void))
```

`edit-menu:preferences-string`

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus edit-menu:preferences-string) ⇒ string
  defaultly returns "(string-constant preferences-menu-item)"
```

`edit-menu:redo-callback`

This method is called when the redo menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:redo-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
(
  (menu evt)
  (let ((edit (get-edit-target-object)))
    (when (and edit (is-a? edit editor<%>))
      (send edit do-edit-operation 'redo)))
  #t)
```

edit-menu:redo-help-string

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus edit-menu:redo-help-string) ⇒ string
  Defaults returns "(string-constant redo-info)"
```

edit-menu:redo-on-demand

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:redo-on-demand item) ⇒ void
  item: menu-item%
  Defaults is this: ( (item) (let* ((editor (get-edit-target-object)) (enable? (and
    editor (is-a? editor editor<\%>)) (send editor can-do-edit-operation? (quote
    redo)))) (send item enable enable?)))
```

edit-menu:redo-string

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus edit-menu:redo-string) ⇒ string
  defaults returns "(string-constant redo-menu-item)"
```

edit-menu:replace-and-find-again-callback

This method is called when the replace-and-find-again menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:replace-and-find-again-callback item
  evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
  Defaults bound to:
  ( (item control) (void))
```

`edit-menu:replace-and-find-again-help-string`

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus edit-menu:replace-and-find-again-help-string) =>
  string
  Defaultly returns "(string-constant replace-and-find-again-info)"
```

`edit-menu:replace-and-find-again-on-demand`

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:replace-and-find-again-on-demand item)
  => void
  item: menu-item%
  Defaultly is this: ( (item) (send item enable (let ((target (get-edit-target-object)))
    (and target (is-a? target editor<\%>))))))
```

`edit-menu:replace-and-find-again-string`

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus edit-menu:replace-and-find-again-string) => string
  defaultly returns "(string-constant replace-and-find-again-menu-item)"
```

`edit-menu:select-all-callback`

This method is called when the select-all menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:select-all-callback item evt) => void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
  Defaultly bound to:
  (
    (menu evt)
    (let ((edit (get-edit-target-object)))
      (when (and edit (is-a? edit editor<%>))
        (send edit do-edit-operation 'select-all)))
    #t)
```

```
edit-menu:select-all-help-string
```

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus edit-menu:select-all-help-string) => string
  Defaults returns "(string-constant select-all-info)"
```

```
edit-menu:select-all-on-demand
```

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus edit-menu:select-all-on-demand item) => void
  item: menu-item%
  Defaults is this: ( (item) (let* ((editor (get-edit-target-object)) (enable? (and
    editor (is-a? editor editor<\%>)) (send editor can-do-edit-operation? (quote
    select-all)))))) (send item enable enable?)))
```

```
edit-menu:select-all-string
```

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus edit-menu:select-all-string) => string
  defaults returns "(string-constant select-all-menu-item)"
```

```
edit-menu:undo-callback
```

This method is called when the undo menu-item of the edit-menu menu is selected.

```
- (send a-frame:standard-menus edit-menu:undo-callback item evt) => void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaults bound to:

```
(
  (menu evt)
  (let ((edit (get-edit-target-object)))
    (when (and edit (is-a? edit editor<%>))
      (send edit do-edit-operation 'undo)))
  #t)
```

```
edit-menu:undo-help-string
```

This result of this method is used as the help string when the `menu-item%` object is created.

- (send `a-frame:standard-menus` `edit-menu:undo-help-string`) ⇒ `string`
 Defaults returns `"(string-constant undo-info)"`

`edit-menu:undo-on-demand`

The menu item's on-demand method calls this method

- (send `a-frame:standard-menus` `edit-menu:undo-on-demand` `item`) ⇒ `void`
`item:` `menu-item%`
 Defaults is this: ((`item`) (let* ((`editor` (`get-edit-target-object`)) (`enable?` (and `editor` (`is-a? editor editor<\%>`) (`send editor can-do-edit-operation?` (`quote undo`)))))) (`send item enable enable?`)))

`edit-menu:undo-string`

The result of this method is the name of this menu.

- (send `a-frame:standard-menus` `edit-menu:undo-string`) ⇒ `string`
 defaults returns `"(string-constant undo-menu-item)"`

`file-menu:after-quit`

This method is called after the addition of the quit menu-item to the file-menu menu. Override it to add additional menus at that point.

- (send `a-frame:standard-menus` `file-menu:after-quit` `menu`) ⇒ `void`
`menu:` (instance (derived-from `menu%`))
 Does nothing.

`file-menu:between-close-and-quit`

This method is called between the addition of the close menu-item and before the addition of the quit menu-item to the file-menu menu. Override it to add additional menus at that point.

- (send `a-frame:standard-menus` `file-menu:between-close-and-quit` `menu`) ⇒ `void`
`menu:` (instance (derived-from `menu%`))
 Does nothing.

file-menu:between-new-and-open

This method is called between the addition of the new menu-item and before the addition of the open menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus file-menu:between-new-and-open menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

file-menu:between-open-and-revert

This method is called between the addition of the open menu-item and before the addition of the revert menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus file-menu:between-open-and-revert menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

file-menu:between-print-and-close

This method is called between the addition of the print menu-item and before the addition of the close menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus file-menu:between-print-and-close menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Adds a separator menu item.

file-menu:between-revert-and-save

This method is called between the addition of the revert menu-item and before the addition of the save menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus file-menu:between-revert-and-save menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

file-menu:between-save-as-and-print

This method is called between the addition of the save-as menu-item and before the addition of the print menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus file-menu:between-save-as-and-print menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

`file-menu:close-callback`

This method is called when the close menu-item of the file-menu menu is selected.

```
- (send a-frame:standard-menus file-menu:close-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
( (item control) (when (can-close?) (on-close) (show #f)) #t)
```

`file-menu:close-help-string`

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus file-menu:close-help-string) ⇒ string
```

Defaultly returns "(string-constant close-info)"

`file-menu:close-on-demand`

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus file-menu:close-on-demand item) ⇒ void
  item: menu-item%
```

Defaultly is this: ((menu-item) (void))

`file-menu:close-string`

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus file-menu:close-string) ⇒ string
```

defaultly returns "(string-constant close-menu-item)"

`file-menu:create-close?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* file-menu:create-close?) ⇒ **boolean**
defaultly returns #t

file-menu:create-new?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* file-menu:create-new?) ⇒ **boolean**
defaultly returns #t

file-menu:create-open-recent?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* file-menu:create-open-recent?) ⇒ **boolean**
defaultly returns #t

file-menu:create-open?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* file-menu:create-open?) ⇒ **boolean**
defaultly returns #t

file-menu:create-print?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:standard-menus* file-menu:create-print?) ⇒ **boolean**
defaultly returns #f

`file-menu:create-quit?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus file-menu:create-quit?) ⇒ boolean
  defaultly returns (not (current-eventspace-has-standard-menus?))
```

`file-menu:create-revert?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus file-menu:create-revert?) ⇒ boolean
  defaultly returns #f
```

`file-menu:create-save-as?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus file-menu:create-save-as?) ⇒ boolean
  defaultly returns #f
```

`file-menu:create-save?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus file-menu:create-save?) ⇒ boolean
  defaultly returns #f
```

`file-menu:get-close-item`

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-close-item) ⇒ (instance menu-item%)
```

file-menu:get-new-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-new-item) ⇒ (instance menu-item%)
```

file-menu:get-open-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-open-item) ⇒ (instance menu-item%)
```

file-menu:get-open-recent-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-open-recent-item) ⇒ (instance menu-item%)
```

file-menu:get-print-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-print-item) ⇒ (instance menu-item%)
```

file-menu:get-quit-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-quit-item) ⇒ (instance menu-item%)
```

file-menu:get-revert-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-revert-item) ⇒ (instance menu-item%)
```

file-menu:get-save-as-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-save-as-item) ⇒ (instance menu-item%)
```

file-menu:get-save-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus file-menu:get-save-item) ⇒ (instance menu-item%)
```

file-menu:new-callback

This method is called when the new menu-item of the file-menu menu is selected.

```
- (send a-frame:standard-menus file-menu:new-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
( (item control) (handler:edit-file #f) #t)
```

file-menu:new-help-string

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus file-menu:new-help-string) ⇒ string
```

Defaultly returns "(string-constant new-info)"

file-menu:new-on-demand

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus file-menu:new-on-demand item) ⇒ void
  item: menu-item%
```

Defaultly is this: ((menu-item) (void))

file-menu:new-string

The result of this method is the name of this menu.

- (send *a-frame:standard-menus* file-menu:new-string) ⇒ string
 defaults returns "(string-constant new-menu-item)"

file-menu:open-callback

This method is called when the open menu-item of the file-menu menu is selected.

- (send *a-frame:standard-menus* file-menu:open-callback *item* *evt*) ⇒ void
item: (instance (derived-from menu-item%))
evt: (instance control-event%)

Defaults bound to:

((*item* *control*) (handler:open-file) #t)

file-menu:open-help-string

This result of this method is used as the help string when the *menu-item%* object is created.

- (send *a-frame:standard-menus* file-menu:open-help-string) ⇒ string
 Defaults returns "(string-constant open-info)"

file-menu:open-on-demand

The menu item's on-demand method calls this method

- (send *a-frame:standard-menus* file-menu:open-on-demand *item*) ⇒ void
item: menu-item%
 Defaults is this: ((*menu-item*) (void))

file-menu:open-recent-callback

This method is called when the open-recent menu-item of the file-menu menu is selected.

- (send *a-frame:standard-menus* file-menu:open-recent-callback *item* *evt*) ⇒ void
item: (instance (derived-from menu-item%))
evt: (instance control-event%)

Defaultly bound to:

```
( (x y) (void))
```

```
file-menu:open-recent-help-string
```

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus file-menu:open-recent-help-string) => string
  Defaultly returns "(string-constant open-recent-info)"
```

```
file-menu:open-recent-on-demand
```

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus file-menu:open-recent-on-demand item) => void
  item: menu-item%
  Defaultly is this: ( (menu) (handler:install-recent-items menu))
```

```
file-menu:open-recent-string
```

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus file-menu:open-recent-string) => string
  defaultly returns "(string-constant open-recent-menu-item)"
```

```
file-menu:open-string
```

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus file-menu:open-string) => string
  defaultly returns "(string-constant open-menu-item)"
```

```
file-menu:print-callback
```

This method is called when the print menu-item of the file-menu menu is selected.

```
- (send a-frame:standard-menus file-menu:print-callback item evt) => void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
( (item control) (void))
```

```
file-menu:print-help-string
```

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus file-menu:print-help-string) ⇒ string
  Defaultly returns "(string-constant print-info)"
```

```
file-menu:print-on-demand
```

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus file-menu:print-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (menu-item) (void))
```

```
file-menu:print-string
```

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus file-menu:print-string) ⇒ string
  defaultly returns "(string-constant print-menu-item)"
```

```
file-menu:quit-callback
```

This method is called when the quit menu-item of the file-menu menu is selected.

```
- (send a-frame:standard-menus file-menu:quit-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
( (item control) (when (exit:user-oks-exit) (exit:exit)))
```

```
file-menu:quit-help-string
```

This result of this method is used as the help string when the `menu-item%` object is created.

- (send *a-frame:standard-menus* `file-menu:quit-help-string`) ⇒ `string`
 Defaults returns "(string-constant *quit-info*)"

`file-menu:quit-on-demand`

The menu item's on-demand method calls this method

- (send *a-frame:standard-menus* `file-menu:quit-on-demand` *item*) ⇒ `void`
item: `menu-item%`
 Defaults is this: ((*menu-item*) (void))

`file-menu:quit-string`

The result of this method is the name of this menu.

- (send *a-frame:standard-menus* `file-menu:quit-string`) ⇒ `string`
 defaults returns "(if (eq? (system-type) (quote windows)) (string-constant *quit-menu-item-windows*) (string-constant *quit-menu-item-others*))"

`file-menu:revert-callback`

This method is called when the revert menu-item of the file-menu menu is selected.

- (send *a-frame:standard-menus* `file-menu:revert-callback` *item* *evt*) ⇒ `void`
item: (instance (derived-from `menu-item%`))
evt: (instance `control-event%`)
 Defaults bound to:
 ((*item* *control*) (void))

`file-menu:revert-help-string`

This result of this method is used as the help string when the `menu-item%` object is created.

- (send *a-frame:standard-menus* `file-menu:revert-help-string`) ⇒ `string`
 Defaults returns "(string-constant *revert-info*)"

`file-menu:revert-on-demand`

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus file-menu:revert-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (menu-item) (void))
```

file-menu:revert-string

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus file-menu:revert-string) ⇒ string
  defaultly returns "(string-constant revert-menu-item)"
```

file-menu:save-as-callback

This method is called when the save-as menu-item of the file-menu menu is selected.

```
- (send a-frame:standard-menus file-menu:save-as-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
  Defaultly bound to:
  ( (item control) (void))
```

file-menu:save-as-help-string

This result of this method is used as the help string when the menu-item% object is created.

```
- (send a-frame:standard-menus file-menu:save-as-help-string) ⇒ string
  Defaultly returns "(string-constant save-as-info)"
```

file-menu:save-as-on-demand

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus file-menu:save-as-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (menu-item) (void))
```

file-menu:save-as-string

The result of this method is the name of this menu.

- (send `a-frame:standard-menus` `file-menu:save-as-string`) ⇒ `string`
 defaults returns "(string-constant save-as-menu-item)"

`file-menu:save-callback`

This method is called when the save menu-item of the file-menu menu is selected.

- (send `a-frame:standard-menus` `file-menu:save-callback` `item` `evt`) ⇒ `void`
`item`: (instance (derived-from `menu-item%`))
`evt`: (instance `control-event%`)

Defaults bound to:

((`item` `control`) (void))

`file-menu:save-help-string`

This result of this method is used as the help string when the `menu-item%` object is created.

- (send `a-frame:standard-menus` `file-menu:save-help-string`) ⇒ `string`
 Defaults returns "(string-constant save-info)"

`file-menu:save-on-demand`

The menu item's on-demand method calls this method

- (send `a-frame:standard-menus` `file-menu:save-on-demand` `item`) ⇒ `void`
`item`: `menu-item%`
 Defaults is this: ((`menu-item`) (void))

`file-menu:save-string`

The result of this method is the name of this menu.

- (send `a-frame:standard-menus` `file-menu:save-string`) ⇒ `string`
 defaults returns "(string-constant save-menu-item)"

`get-checkable-menu-item%`

The result of this method is used as the class for creating checkable menu items in this class (see `frame:standard-menus%` for a list).

- (send *a-frame:standard-menus* get-checkable-menu-item%) ⇒ (derived-from *checkable-menu-item%*)
defaultly returns *menu:can-restore-checkable-menu-item%*.

get-edit-menu

Returns the edit menu See also *get-menu%*

- (send *a-frame:standard-menus* get-edit-menu) ⇒ (instance (derived-from *menu%*))

get-file-menu

Returns the file menu See also *get-menu%*

- (send *a-frame:standard-menus* get-file-menu) ⇒ (instance (derived-from *menu%*))

get-help-menu

Returns the help menu See also *get-menu%*

- (send *a-frame:standard-menus* get-help-menu) ⇒ (instance (derived-from *menu%*))

get-menu-item%

The result of this method is used as the class for creating the menu items in this frame (see *frame:standard-menus%* for a list).

- (send *a-frame:standard-menus* get-menu-item%) ⇒ (derived-from *menu-item%*)
defaultly returns *menu:can-restore-menu-item%*.

get-menu%

The result of this method is used as the class for creating the result of these methods: *get-file-menu*, *get-edit-menu*, *get-help-menu*.

- (send *a-frame:standard-menus* get-menu%) ⇒ (derived-from *menu:can-restore-underscore-menu%*)
defaultly returns *menu%*

`help-menu:about-callback`

This method is called when the about menu-item of the help-menu menu is selected.

```
- (send a-frame:standard-menus help-menu:about-callback item evt) ⇒ void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Defaultly bound to:

```
( (item control) (void))
```

`help-menu:about-help-string`

This result of this method is used as the help string when the `menu-item%` object is created.

```
- (send a-frame:standard-menus help-menu:about-help-string) ⇒ string
  Defaultly returns "(string-constant about-info)"
```

`help-menu:about-on-demand`

The menu item's on-demand method calls this method

```
- (send a-frame:standard-menus help-menu:about-on-demand item) ⇒ void
  item: menu-item%
  Defaultly is this: ( (menu-item) (void))
```

`help-menu:about-string`

The result of this method is the name of this menu.

```
- (send a-frame:standard-menus help-menu:about-string) ⇒ string
  defaultly returns "(string-constant about-menu-item)"
```

`help-menu:after-about`

This method is called after the addition of the about menu-item to the help-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus help-menu:after-about menu) ⇒ void
  menu: (instance (derived-from menu%))
```

Does nothing.

help-menu:before-about

This method is called before the addition of the about menu-item to the help-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:standard-menus help-menu:before-about menu) => void
  menu: (instance (derived-from menu%))
```

Does nothing.

help-menu:create-about?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:standard-menus help-menu:create-about?) => boolean
  defaultly returns #f
```

help-menu:get-about-item

This method returns the `menu-item%` that corresponds to this menu item.

```
- (send a-frame:standard-menus help-menu:get-about-item) => (instance menu-item%)
```

12.16 frame:standard-menus-mixin

Domain: `frame:basic<%>`

Implements: `frame:basic<%>`

Implements: `frame:standard-menus<%>`

The result of this mixin implements `frame:standard-menus<%>`.

`on-close` (*augments, and augmentable only*)

Called just before the window is closed (e.g., by the window manager). This method is *not* called by `show`.

See also `can-close?`.

- (send a-frame:standard-menus-mixin on-close void
Removes the preferences callbacks for the menu items

12.17 frame:editor<%>

Extends: `frame:standard-menus<%>`

Frame classes matching this interface support embedded editors.

`get-canvas`

Returns the canvas used to display the `editor<%>` in this frame.

- (send a-frame:editor get-canvas) ⇒ (instance (derived-from `canvas%`))

`get-canvas<%>`

The result of this method is used to guard the result of the `get-canvas%` method.

- (send a-frame:editor get-canvas<%>) ⇒ (instance `canvas:basic%`)

`get-canvas%`

The result of this method is used to create the canvas for the `editor<%>` in this frame.

- (send a-frame:editor get-canvas%) ⇒ (derived-from `editor-canvas%`)
Returns `editor-canvas%`.

`get-editor`

Returns the editor in this frame.

- (send a-frame:editor get-editor) ⇒ (instance (implements `editor<%>`))

`get-editor<%>`

The result of this method is used by `make-editor` to check that `get-editor%` is returning a reasonable editor.

- (send *a-frame:editor* get-editor<%>) ⇒ interface

Returns *editor<%>*.

get-editor%

The result of this class is used to create the *editor<%>* in this frame.

Override this method to specify a different editor class.

- (send *a-frame:editor* get-editor%) ⇒ (implements *editor<%>*)

get-entire-label

This method returns the entire label for the frame. See also *set-label* and *set-label-prefix*.

- (send *a-frame:editor* get-entire-label) ⇒ string

get-label-prefix

This returns the prefix for the frame's label.

- (send *a-frame:editor* get-label-prefix) ⇒ string

make-editor

This method is called to create the editor in this frame. It calls *get-editor<%>* and uses that interface to make sure the result of *get-editor%* is reasonable.

- (send *a-frame:editor* make-editor) ⇒ (instance (implements *editor<%>*))

Calls (make-object *get-editor%*).

revert

- (send *a-frame:editor* revert) ⇒ void

Loads the most recently saved version of the file to the disk. If the *editor<%>* is a *text%*, the start and end positions are restored.

save

Saves the file being edited, possibly calling `save-as` if the editor has no filename yet.

```
- (send a-frame:editor save format) ⇒ boolean
  format = 'same: (union 'guess 'standard 'text 'text-force-cr 'same 'copy)
```

Returns #f if the user cancels this operation (only possible when the file has not been saved before and the user is prompted for a new filename) and returns #t if not.

save-as

Queries the user for a file name and saves the file with that name.

```
- (send a-frame:editor save-as format) ⇒ boolean
  format = 'same: (union 'guess 'standard 'text 'text-force-cr 'same 'copy)
```

Returns #f if the user cancels the file-choosing dialog and returns #t otherwise.

set-label-prefix

Sets the prefix for the label of the frame.

```
- (send a-frame:editor set-label-prefix prefix) ⇒ void
  prefix: string
```

12.18 frame:editor-mixin

Domain: `frame:standard-menus<%>`

Implements: `frame:editor<%>`

Implements: `frame:standard-menus<%>`

This mixin adds functionality to support an `editor<%>` in the frame. This includes management of the title, implementations of some of the menu items, a reasonable initial size, and access to the `editor<%>` itself.

The size of this frame will be either 600 by 650 or 65 less than the width and height of the screen, whichever is smaller.

```
- init args: (filename _)
  filename: string?
```

can-close? (*augments, and augmentable only*)

Called just before the window might be closed (e.g., by the window manager). If #f is returned, the window is not closed, otherwise `on-close` is called and the window is closed (i.e., the window is hidden, like calling `show` with #f).

This method is *not* called by `show`.

```
- (send a-frame:editor-mixin can-close? void
  Calls the editor:basic<%>'s method can-close?.
```

edit-menu:between-select-all-and-find

This method is called between the addition of the select-all menu-item and before the addition of the find menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-frame:editor-mixin edit-menu:between-select-all-and-find edit-menu void
  edit-menu: (instance menu%)
  Adds a menu item for toggling auto-wrap in the focused text.
```

editing-this-file?

Indicates if this frame contains this buffer (and can edit that file).

```
- (send a-frame:editor-mixin editing-this-file? filename boolean
  filename: path
  Returns #t if the filename is the file that this frame is editing.
```

file-menu:create-print?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:editor-mixin file-menu:create-print? boolean
  returns #t
```

file-menu:create-revert?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:editor-mixin file-menu:create-revert? boolean
  returns #t
```

file-menu:create-save-as?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:editor-mixin file-menu:create-save-as? boolean
  returns #t
```

file-menu:create-save?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:editor-mixin file-menu:create-save? boolean
  returns #t
```

file-menu:print-callback

This method is called when the print menu-item of the file-menu menu is selected.

```
- (send a-frame:editor-mixin file-menu:print-callback item evt void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Calls the `print` method of `editor<%>` with the default arguments, except that the `output-mode` argument is the result of calling `preferences:get` with `'framework:print-output-mode`.

file-menu:revert-callback

This method is called when the revert menu-item of the file-menu menu is selected.

```
- (send a-frame:editor-mixin file-menu:revert-callback item evt void
  item: (instance (derived-from menu-item%))
  evt: (instance control-event%)
```

Informs the user that this action is not undoable and, if they still want to continue, calls `revert`.

file-menu:revert-on-demand

The menu item's on-demand method calls this method

- (send *a-frame:editor-mixin* file-menu:revert-on-demand void
Disables the menu item when the editor is locked.

file-menu:save-as-callback

This method is called when the save-as menu-item of the file-menu menu is selected.

- (send *a-frame:editor-mixin* file-menu:save-as-callback *item* *evt* void
item: (instance (derived-from menu-item%))
evt: (instance control-event%)
Prompts the user for a file name and uses that filename to save the buffer. Calls `save-as` with no arguments.

file-menu:save-callback

This method is called when the save menu-item of the file-menu menu is selected.

- (send *a-frame:editor-mixin* file-menu:save-callback *item* *evt* void
item: (instance (derived-from menu-item%))
evt: (instance control-event%)
Saves the file in the editor.

get-filename

This returns the filename that the frame is currently being saved as, or #f if there is no appropriate filename.

- (send *a-frame:editor-mixin* get-filename (union #f string)
Returns the filename in the editor returned by `get-editor`.

get-label

Gets a window's label, if any. Control windows generally display their label in some way. Frames and dialogs display their label as a window title. Panels do not display their label, but the label can be used for identification purposes. Messages, buttons, and check boxes can have bitmap labels (only when they are created with bitmap labels), but all other windows have string labels. In addition, a message label can be an icon symbol 'app, 'caution, or 'stop.

The label string may contain ampersands ("&"), which serve as keyboard navigation annotations for controls under Windows and X. The ampersands are not part of the displayed label of a control; instead, ampersands are removed in the displayed label (under all platforms), and any character preceding an ampersand is underlined (Windows and X) indicating that the character is a mnemonic for the control. Double ampersands are converted into a single ampersand (with no displayed underline). See also `on-traverse-char`.

If the window does not have a label, #f is returned.

- (send *a-frame:editor-mixin* get-label string
Returns the portion of the label after the hyphen. See also `get-entire-label`.

help-menu:about-callback

This method is called when the about menu-item of the help-menu menu is selected.

- (send *a-frame:editor-mixin* help-menu:about-callback *item* *evt* void
item: (instance (derived-from `menu-item%`))
evt: (instance `control-event%`)
Calls `message-box` with a message welcoming the user to the application named by `application:current-app-name`

help-menu:about-string

The result of this method is the name of this menu.

- (send *a-frame:editor-mixin* help-menu:about-string string
Returns the result of (`application:current-app-name`)

help-menu:create-about?

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:editor-mixin* help-menu:create-about? boolean
returns #t

on-close (*augments, and augmentable only*)

Called just before the window is closed (e.g., by the window manager). This method is *not* called by `show`.

See also `can-close?`.

- (send *a-frame:editor-mixin* on-close void
Calls the `editor:basic<%>`'s method `on-close`.

set-label

Sets a window's label. The window's natural minimum size might be different after the label is changed, but the window's minimum size is not recomputed.

See `get-label` for more information.

```
- (send a-frame:editor-mixin set-label l void
  l : string (up to 200 characters)
```

Sets the label, but preserves the label's prefix. See also `set-label-prefix`.

12.19 frame:open-here<%>

Extends: `frame:editor<%>`

Frames implementing this mixin can change the file they are displaying.

The frame is only re-used when the `'framework:open-here?` preference is set (see `preferences:get` and `preferences:set` for details on preferences).

The `frame:open-here-mixin` implements this interface.

`get-open-here-editor`

When the user switches the visible file in this frame, the of this method is the editor that gets switched.

```
- (send a-frame:open-here get-open-here-editor) => (is-a?/c editor|%¿)
```

Defaults returns the result of `get-editor`.

`open-here`

```
- (send a-frame:open-here open-here filename) => void
  filename : string
```

Opens `filename` in the current frame, possibly prompting the user about saving a file (in which case the frame might not get switched).

12.20 frame:open-here-mixin

Domain: `frame:editor<%>`

Implements: `frame:editor<%>`

Implements: `frame:open-here<%>`

Provides an implementation of `frame:open-here<%>`

`file-menu:new-callback`

This method is called when the new menu-item of the file-menu menu is selected.

```
- (send a-frame:open-here-mixin file-menu:new-callback item evt void
    item: (instance (derived-from menu-item%))
    evt: (instance control-event%))
```

When the preference 'framework:open-here? preference is set, this method prompts the user, asking if they would like to create a new frame, or just clear out this one. If they clear it out and the file hasn't been saved, they are asked about saving.

`file-menu:new-on-demand`

The menu item's on-demand method calls this method

```
- (send a-frame:open-here-mixin file-menu:new-on-demand item void
    item: (is-a?/c menu-item%))
```

Sets the label of `item` to "New..." if the preference 'framework:open-here? is set.

`file-menu:open-on-demand`

The menu item's on-demand method calls this method

```
- (send a-frame:open-here-mixin file-menu:open-on-demand item void
    item: (is-a?/c menu-item%))
```

Sets the label of `item` to "Open Here..." if the preference 'framework:open-here? is set.

`on-activate`

Called when a window is *activated* or *deactivated*. A top-level window is activated when the keyboard focus moves from outside the window to the window or one of its children. It is deactivated when the focus moves back out of the window. Under Mac OS X, a child of a floating frames can have the focus instead of a child of the active non-floating frame; in other words, floating frames act as an extension of the active non-frame for keyboard focus.

The method's argument is `#t` when the window is activated, `#f` when it is deactivated.

```
- (send a-frame:open-here-mixin on-activate on? void
  on?: boolean
```

When `on?` is `#t`, calls `set-open-here-frame` with this.

`on-close` (*augments, and augmentable only*)

Called just before the window is closed (e.g., by the window manager). This method is *not* called by `show`.

See also `can-close?`.

```
- (send a-frame:open-here-mixin on-close void
```

Calls `set-open-here-frame` with `#f` if the result of `get-open-here-frame` is `eq?` to this.

12.21 frame:text<%>

Extends: `frame:editor<%>`

Frames matching this interface provide support for `text%`.

12.22 frame:text-mixin

Domain: `frame:editor<%>`

Implements: `frame:editor<%>`

Implements: `frame:text<%>`

This mixins adds support for `text%` in the frame.

The result of this mixin uses the same initialization arguments as the mixin's argument.

`get-editor<%>`

The result of this method is used by `make-editor` to check that `get-editor%` is returning a reasonable editor.

```
- (send a-frame:text-mixin get-editor<%> interface
```

Returns (class->interface `text%`).

`get-editor%`

The result of this class is used to create the `editor<%>` in this frame.

Override this method to specify a different editor class.

```
- (send a-frame:text-mixin get-editor% (implements editor<%>)  
  Returns text:keymap%.
```

12.23 `frame:pasteboard<%>`

Extends: `frame:editor<%>`

Frames matching this interface provide support for `pasteboard%`s.

12.24 `frame:pasteboard-mixin`

Domain: `frame:editor<%>`

Implements: `frame:editor<%>`

Implements: `frame:pasteboard<%>`

This mixin provides support for pasteboards in a frame.

The result of this mixin uses the same initialization arguments as the mixin's argument.

`get-editor<%>`

The result of this method is used by `make-editor` to check that `get-editor%` is returning a reasonable editor.

```
- (send a-frame:pasteboard-mixin get-editor<%> interface  
  Returns (class->interface pasteboard%).
```

`get-editor%`

The result of this class is used to create the `editor<%>` in this frame.

Override this method to specify a different editor class.

```
- (send a-frame:clipboard-mixin get-editor% (implements editor<%>))
  Returns clipboard:keymap%.
```

12.25 frame:delegate<%>

Extends: `frame:status-line<%>`

Extends: `frame:text<%>`

Frames that implement this interface provide a 20,000 feet overview of the text in the main editor. The term **delegate** in these method descriptions refers to the original editor and the term **delegatee** refers to the editor showing the 20,000 feet overview.

delegate-moved

This method is called when the visible region of the delegate editor changes, so that the blue region in the delegatee is updated.

```
- (send a-frame:delegate delegate-moved) ⇒ void
```

delegated-text-shown?

Returns #t if the delegate is visible, and #f if it isn't.

```
- (send a-frame:delegate delegated-text-shown?) ⇒ boolean
```

get-delegated-text

Returns the delegate text.

```
- (send a-frame:delegate get-delegated-text) ⇒ (instanceof (implements text:delegate<%>))
```

hide-delegated-text

Hides the delegated text.

When the delegated text is hidden, it is not being updated. This is accomplished by calling the `set-delegate` method of `get-editor` with `#f`.

See also `show-delegated-text`

```
- (send a-frame:delegate hide-delegated-text) ⇒ void
```

```
show-delegated-text
```

Makes the delegated text visible.

When the delegated text is shown, the `set-delegate` method of `get-delegated-text` is called with the text to delegate messages to.

See also `hide-delegated-text`.

```
- (send a-frame:delegate show-delegated-text) ⇒ void
```

12.26 frame:delegate-mixin

Domain: `frame:status-line<%>`

Domain: `frame:text<%>`

Implements: `frame:status-line<%>`

Implements: `frame:text<%>`

Implements: `frame:delegate<%>`

Adds support for a 20,000-foot view via `text:delegate<%>` and `text:delegate-mixin`

```
get-editor<%>
```

The result of this method is used by `make-editor` to check that `get-editor%` is returning a reasonable editor.

```
- (send a-frame:delegate-mixin get-editor<%> interface
  Returns text:delegate<%>.
```

```
get-editor%
```

The result of this class is used to create the `editor<%>` in this frame.

Override this method to specify a different editor class.

```
- (send a-frame:delegate-mixin get-editor% (implements text:delegate<%>))
  returns the super result, with the text:delegate-mixin mixed in.
```

`make-root-area-container`

Override this method to insert a panel in between the panel used by the clients of this frame and the frame itself. For example, to insert a status line panel override this method with something like this:

```
(class ...
  ...
  (rename [super-make-root-area-container make-root-area-container])
  (field
    [status-panel #f])
  (define/override (make-root-area-container cls parent)
    (set! status-panel
          (super-make-root-area-container vertical-panel% parent))
    (let ([root (make-object cls status-panel)]

          ; ... add other children to status-panel ...

      root))
  ...
```

In this example, `status-panel` will contain a root panel for the other classes, and whatever panels are needed to display status information.

The searching frame is implemented using this method.

```
- (send a-frame:delegate-mixin make-root-area-container class parent (implements
  panel%)
  class: (derived-from panel%)
  parent: (instanceof (derived-from panel%))
  adds a panel outside to hold the delegate editor-canvas% and text%.
```

12.27 frame:searchable<%>

Extends: `frame:basic<%>`

Frames that implement this interface support searching.

`can-replace?`

Returns `#t` if a replace command would succeed.

```
- (send a-frame:searchable can-replace?) ⇒ boolean
```

Defaultly is `#t` when the selected text in the result of `get-text-to-search` is the same as the text in the find text.

`get-text-to-search`

Override this method to specify which text to search.

```
- (send a-frame:searchable get-text-to-search) ⇒ (instance (derived-from text%))
```

Returns the result of `get-editor`.

`hide-search`

This method hides the searching information on the bottom of the frame.

```
- (send a-frame:searchable hide-search) ⇒ void
```

`move-to-search-or-reverse-search`

This method moves the focus to the text that contains the search string, or if the focus is there already, performs a reverse search.

It returns `void` if the focus was not to the search text, otherwise it returns a boolean indicating the success of the search.

```
- (send a-frame:searchable move-to-search-or-reverse-search) ⇒ (union boolean void)
```

`move-to-search-or-search`

This method moves the focus to the text that contains the search string, or if the focus is there already, performs a forward search.

It returns `void` if the focus was not to the search text, otherwise it returns a boolean indicating the success of the search.

```
- (send a-frame:searchable move-to-search-or-search) ⇒ (union boolean void)
```

replace

If the selected text matches the search string, this method replaces the text with the contents of the replace text. If the replace was successful, #t is returned. Otherwise, #f is returned.

```
- (send a-frame:searchable replace) ⇒ boolean
```

replace-all

Loops through the text from the current position to the end, replacing all occurrences of the search string with the contents of the replace edit. Only searches forward, does not loop around to the beginning of the text.

```
- (send a-frame:searchable replace-all) ⇒ void
```

replace&search

Calls `replace` and if it returns #t, calls `search-again`.

```
- (send a-frame:searchable replace&search) ⇒ boolean
```

search-again

Searches for the text in the search edit in the result of `get-text-to-search`.

```
- (send a-frame:searchable search-again direction beep?) ⇒ boolean
  direction = previous searching direction: 'forward or 'backward
  beep? = #t : bool
```

Returns #t if the text is found and sets the selection to the found text. If the text is not found it returns #f.

set-search-direction

Sets the direction that future searches will be performed.

```
- (send a-frame:searchable set-search-direction dir) ⇒ void
  dir: (union -1 1)
```

If `dir` is 1 searches will be performed forwards and if `dir` is -1 searches will be performed backwards.

toggle-search-focus

Toggles the keyboard focus between the searching edit, the replacing edit and the result of `get-text-to-search`.

```
- (send a-frame:searchable toggle-search-focus) ⇒ void
```

`unhide-search`

When the searching sub window is hidden, makes it visible.

```
- (send a-frame:searchable unhide-search) ⇒ void
```

12.28 `frame:searchable-mixin`

Domain: `frame:standard-menus<%>`

Implements: `frame:searchable<%>`

Implements: `frame:standard-menus<%>`

This mixin adds support for searching in the `editor<%>` in this frame.

The result of this mixin uses the same initialization arguments as the mixin's argument.

`edit-menu:create-find-again?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:searchable-mixin edit-menu:create-find-again? boolean
  returns #t
```

`edit-menu:create-find?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-frame:searchable-mixin edit-menu:create-find? boolean
  returns #t
```

`edit-menu:create-replace-and-find-again?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

- (send *a-frame:searchable-mixin* `edit-menu:create-replace-and-find-again?` **boolean** returns #t

`edit-menu:find-again-callback`

This method is called when the find-again menu-item of the edit-menu menu is selected.

- (send *a-frame:searchable-mixin* `edit-menu:find-again-callback` **boolean**
Returns #t, and searches for the same text that was last searched for in the text.

`edit-menu:find-callback`

This method is called when the find menu-item of the edit-menu menu is selected.

- (send *a-frame:searchable-mixin* `edit-menu:find-callback` *item* *evt* **void**
item: (instance (derived-from `menu-item%`))
evt: (instance `control-event%`)
Calls `move-to-search-or-search`.

`edit-menu:replace-and-find-again-callback`

This method is called when the replace-and-find-again menu-item of the edit-menu menu is selected.

- (send *a-frame:searchable-mixin* `edit-menu:replace-and-find-again-callback` **boolean**
Returns #t, and if the selected text matches the current text in the find box, replaces it with the contents of the replace box and searches for the next occurrence of the text in the find box.

`edit-menu:replace-and-find-again-on-demand`

The menu item's on-demand method calls this method

- (send *a-frame:searchable-mixin* `edit-menu:replace-and-find-again-on-demand` *item* **void**
item: `menu-item%`
Disables *item* when `can-replace?` returns #f and enables it when that method returns #t.

make-root-area-container

Override this method to insert a panel in between the panel used by the clients of this frame and the frame itself. For example, to insert a status line panel override this method with something like this:

```
(class ...
  ...
  (rename [super-make-root-area-container make-root-area-container])
  (field
    [status-panel #f])
  (define/override (make-root-area-container cls parent)
    (set! status-panel
      (super-make-root-area-container vertical-panel% parent))
    (let ([root (make-object cls status-panel)])

      ; ... add other children to status-panel ...

      root))
  ...)
```

In this example, status-panel will contain a root panel for the other classes, and whatever panels are needed to display status information.

The searching frame is implemented using this method.

```
- (send a-frame:searchable-mixin make-root-area-container (implements area-container<%>)
  Builds a panel for the searching information.
```

on-activate

Called when a window is *activated* or *deactivated*. A top-level window is activated when the keyboard focus moves from outside the window to the window or one of its children. It is deactivated when the focus moves back out of the window. Under Mac OS X, a child of a floating frames can have the focus instead of a child of the active non-floating frame; in other words, floating frames act as an extension of the active non-frame for keyboard focus.

The method's argument is #t when the window is activated, #f when it is deactivated.

```
- (send a-frame:searchable-mixin on-activate active? void
  active?: boolean
  When the frame is activated, searches will take place in this frame.
```

on-close (*augments, and augmentable only*)

Called just before the window is closed (e.g., by the window manager). This method is *not* called by `show`.

See also `can-close?`.

- (send *a-frame:searchable-mixin* on-close void
Cleans up after the searching frame.

12.29 frame:searchable-text<%>

Extends: `frame:searchable<%>`

Extends: `frame:text<%>`

12.30 frame:searchable-text-mixin

Domain: `frame:searchable<%>`

Domain: `frame:text<%>`

Implements: `frame:searchable-text<%>`

Implements: `frame:searchable<%>`

Implements: `frame:text<%>`

`get-editor<%>`

The result of this method is used by `make-editor` to check that `get-editor%` is returning a reasonable editor.

- (send *a-frame:searchable-text-mixin* get-editor<%> (implements `editor<%>`)
Returns `text:searching<%>`.

`get-editor%`

The result of this class is used to create the `editor<%>` in this frame.

Override this method to specify a different editor class.

- (send *a-frame:searchable-text-mixin* get-editor% (implements `editor<%>`)
Returns `text:searching%`.

get-text-to-search

Override this method to specify which text to search.

```
- (send a-frame:searchable-text-mixin get-text-to-search (instanceof text%))
```

Returns the result of `get-editor`.

12.31 frame:basic% = (frame:register-group-mixin (frame:basic-mixin frame%))

```
frame:basic% = (frame:register-group-mixin (frame:basic-mixin frame%))
```

```
- (new frame:basic% (label _) [(parent _)] [(width _)] [(height _)] [(x _)] [(y _)] [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)]) => frame:basic% object
```

label : string (up to 200 characters)

parent = #f : frame% object or #f

width = #f : exact integer in [0, 10000] or #f

height = #f : exact integer in [0, 10000] or #f

x = #f : exact integer in [-10000, 10000] or #f

y = #f : exact integer in [-10000, 10000] or #f

style = null : list of symbols in '(no-resize-border no-caption no-system-menu mdi-parent mdi-child toolbar-button hide-menu-bar float metal)

enabled = #t : boolean

border = 0 : exact integer in [0, 1000]

spacing = 0 : exact integer in [0, 1000]

alignment = '(center top) : two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom

min-width = 0 : exact integer in [0, 10000]

min-height = 0 : exact integer in [0, 10000]

stretchable-width = #t : boolean

stretchable-height = #t : boolean

Passes all arguments to `super-init`.

12.32 frame:size-pref% = (frame:size-pref-mixin frame:basic%)

```
frame:size-pref% = (frame:size-pref-mixin frame:basic%)
```

```
- (new frame:size-pref% (label _) [(parent _)] [(x _)] [(y _)] [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)] (size-preferences-key _)) => frame:size-pref% object
```

label : string (up to 200 characters)

parent = #f : frame% object or #f

x = #f : exact integer in [-10000, 10000] or #f

y = #f : exact integer in [-10000, 10000] or #f

```

style = null: list of symbols in '(no-resize-border no-caption no-system-menu
    mdi-parent mdi-child toolbar-button hide-menu-bar float
    metal)
enabled = #t: boolean
border = 0: exact integer in [0, 1000]
spacing = 0: exact integer in [0, 1000]
alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
min-width = 0: exact integer in [0, 10000]
min-height = 0: exact integer in [0, 10000]
stretchable-width = #t: boolean
stretchable-height = #t: boolean
size-preferences-key: symbol

```

Passes all arguments to super-init.

12.33 frame:info% = (frame:info-mixin frame:basic%)

```
frame:info% = (frame:info-mixin frame:basic%)
```

```
- (new frame:info% (label _) [(parent _)] [(width _)] [(height _)] [(x _)] [(y _)]
  [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width
  _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)]) => frame:info%
object
```

```

label: string (up to 200 characters)
parent = #f: frame% object or #f
width = #f: exact integer in [0, 10000] or #f
height = #f: exact integer in [0, 10000] or #f
x = #f: exact integer in [-10000, 10000] or #f
y = #f: exact integer in [-10000, 10000] or #f
style = null: list of symbols in '(no-resize-border no-caption no-system-menu
    mdi-parent mdi-child toolbar-button hide-menu-bar float
    metal)
enabled = #t: boolean
border = 0: exact integer in [0, 1000]
spacing = 0: exact integer in [0, 1000]
alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
min-width = 0: exact integer in [0, 10000]
min-height = 0: exact integer in [0, 10000]
stretchable-width = #t: boolean
stretchable-height = #t: boolean

```

Passes all arguments to super-init.

12.34 frame:text-info% = (frame:text-info-mixin frame:info%)

```
frame:text-info% = (frame:text-info-mixin frame:info%)
```

```
- (new frame:text-info% (label _) [(parent _)] [(width _)] [(height _)] [(x _)]
  [(y _)] [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width
  _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)]) => frame:text-info%
object
```

```
label: string (up to 200 characters)
```

12.35. `frame:pasteboard-info%` = (`frame:pasteboard-info-mixin` `frame:text-info%`) *Frame*

```
parent = #f: frame% object or #f
width = #f: exact integer in [0, 10000] or #f
height = #f: exact integer in [0, 10000] or #f
x = #f: exact integer in [-10000, 10000] or #f
y = #f: exact integer in [-10000, 10000] or #f
style = null: list of symbols in '(no-resize-border no-caption no-system-menu
      mdi-parent mdi-child toolbar-button hide-menu-bar float
      metal)
enabled = #t: boolean
border = 0: exact integer in [0, 1000]
spacing = 0: exact integer in [0, 1000]
alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
min-width = 0: exact integer in [0, 10000]
min-height = 0: exact integer in [0, 10000]
stretchable-width = #t: boolean
stretchable-height = #t: boolean
```

Passes all arguments to `super-init`.

12.35 `frame:pasteboard-info%` = (`frame:pasteboard-info-mixin` `frame:text-info%`)

```
frame:pasteboard-info% = (frame:pasteboard-info-mixin frame:text-info%)
```

```
- (new frame:pasteboard-info% (label _) [(parent _)] [(width _)] [(height _)] [(x
_) [(y _)] [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)]
[(min-width _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height
_)]) => frame:pasteboard-info% object
  label: string (up to 200 characters)
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
      mdi-parent mdi-child toolbar-button hide-menu-bar float
      metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to `super-init`.

12.36 `frame:status-line%` = (`frame:status-line-mixin` `frame:text-info%`)

```
frame:status-line% = (frame:status-line-mixin frame:text-info%)
```

12. *Filament*7. `frame:standard-menus% = (frame:standard-menus-mixin frame:status-line%)`

```
- (new frame:status-line% (label _) [(parent _)] [(width _)] [(height _)] [(x _)]
  [(y _)] [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width
  _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)]) => frame:status-line%
object
  label : string (up to 200 characters)
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
    mdi-parent mdi-child toolbar-button hide-menu-bar float
    metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to `super-init`.

12.37 `frame:standard-menus% = (frame:standard-menus-mixin frame:status-line%)`

```
frame:standard-menus% = (frame:standard-menus-mixin frame:status-line%)
```

```
- (new frame:standard-menus% (label _) [(parent _)] [(width _)] [(height _)] [(x
  _)] [(y _)] [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)]
  [(min-width _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height
  _)]) => frame:standard-menus% object
  label : string (up to 200 characters)
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
    mdi-parent mdi-child toolbar-button hide-menu-bar float
    metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to `super-init`.

12.38 frame:editor% = (frame:editor-mixin frame:standard-menus%)

```
frame:editor% = (frame:editor-mixin frame:standard-menus%)
```

```
- (new frame:editor% [(parent -)] [(width -)] [(height -)] [(x -)] [(y -)] [(style
-)] [(enabled -)] [(border -)] [(spacing -)] [(alignment -)] [(min-width -)] [(min-height
-)] [(stretchable-width -)] [(stretchable-height -)] (filename -)) => frame:editor%
object
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
mdi-parent mdi-child toolbar-button hide-menu-bar float
metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
  filename: string?
```

Passes all arguments to super-init.

12.39 frame:open-here% = (frame:open-here-mixin frame:editor%)

```
frame:open-here% = (frame:open-here-mixin frame:editor%)
```

```
- (new frame:open-here% [(parent -)] [(width -)] [(height -)] [(x -)] [(y -)] [(style
-)] [(enabled -)] [(border -)] [(spacing -)] [(alignment -)] [(min-width -)] [(min-height
-)] [(stretchable-width -)] [(stretchable-height -)] (filename -)) => frame:open-here%
object
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
mdi-parent mdi-child toolbar-button hide-menu-bar float
metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
  filename: string?
```

Passes all arguments to super-init.

12.40 frame:text% = (frame:text-mixin frame:open-here%)

```
frame:text% = (frame:text-mixin frame:open-here%)
```

```
- (new frame:text% [(parent -)] [(width -)] [(height -)] [(x -)] [(y -)] [(style
-) ] [(enabled -)] [(border -)] [(spacing -)] [(alignment -)] [(min-width -)] [(min-height
-) ] [(stretchable-width -)] [(stretchable-height -)] (filename -) => frame:text%
object
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
mdi-parent mdi-child toolbar-button hide-menu-bar float
metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
  filename: string?
```

Passes all arguments to super-init.

12.41 frame:searchable% = (frame:searchable-mixin (frame:searchable-text-mixin frame:text%))

```
frame:searchable% = (frame:searchable-mixin (frame:searchable-text-mixin frame:text%))
```

```
- (new frame:searchable% [(parent -)] [(width -)] [(height -)] [(x -)] [(y -)] [(style
-) ] [(enabled -)] [(border -)] [(spacing -)] [(alignment -)] [(min-width -)] [(min-height
-) ] [(stretchable-width -)] [(stretchable-height -)] (filename -) => frame:searchable%
object
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
mdi-parent mdi-child toolbar-button hide-menu-bar float
metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
```

```

min-width=0: exact integer in [0, 10000]
min-height=0: exact integer in [0, 10000]
stretchable-width=#t: boolean
stretchable-height=#t: boolean
filename: string?

```

Passes all arguments to super-init.

12.42 frame:delegate% = (frame:delegate-mixin frame:searchable%)

```
frame:delegate% = (frame:delegate-mixin frame:searchable%)
```

```

- (new frame:delegate% [(parent _)] [(width _)] [(height _)] [(x _)] [(y _)] [(style
_) ] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width _)] [(min-height
_) ] [(stretchable-width _)] [(stretchable-height _)] (filename _) => frame:delegate%
object
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f
  style = null: list of symbols in '(no-resize-border no-caption no-system-menu
mdi-parent mdi-child toolbar-button hide-menu-bar float
metal)
  enabled = #t: boolean
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
  filename: string?

```

Passes all arguments to super-init.

12.43 frame:pasteboard% = (frame:pasteboard-mixin frame:open-here%)

```
frame:pasteboard% = (frame:pasteboard-mixin frame:open-here%)
```

```

- (new frame:pasteboard% [(parent _)] [(width _)] [(height _)] [(x _)] [(y _)] [(style
_) ] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width _)] [(min-height
_) ] [(stretchable-width _)] [(stretchable-height _)] (filename _) => frame:pasteboard%
object
  parent = #f: frame% object or #f
  width = #f: exact integer in [0, 10000] or #f
  height = #f: exact integer in [0, 10000] or #f
  x = #f: exact integer in [-10000, 10000] or #f
  y = #f: exact integer in [-10000, 10000] or #f

```

```
style = null : list of symbols in '(no-resize-border no-caption no-system-menu
    mdi-parent mdi-child toolbar-button hide-menu-bar float
    metal)
enabled = #t : boolean
border = 0 : exact integer in [0, 1000]
spacing = 0 : exact integer in [0, 1000]
alignment = '(center top) : two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
min-width = 0 : exact integer in [0, 10000]
min-height = 0 : exact integer in [0, 10000]
stretchable-width = #t : boolean
stretchable-height = #t : boolean
filename : string?
```

Passes all arguments to super-init.

13. Group

A *frame group* associates a group of frames together. There is one frame group in mred, called `group:the-frame-group`, which is an object of the `group:%` class.

The frame group manages the windows menu. It also and enables the close menu item on each frame, when there is more than one frame in the group, and disables the close menu item when there is only one frame in the frame group.

- `group:%`

13.1 `group:%`

This class manages a group of frames matching the `frame:basic<%>` interface. There is one instance created by the framework, returned by the function `group:get-the-frame-group` and every frame that was constructed with `frame:basic-mixin` adds itself to the result of `group:get-the-frame-group`.

`can-close-all?`

Call this method to make sure that closing all of the frames in the frame groups is permitted by the user. The function `on-close-all` is expected to be called just after this method is called.

- (send *a-group:* `can-close-all?`) ⇒ void
Calls the `can-close?` method of each frame in the group.

`clear`

This removes all of the frames in the group. It does not close the frames. See also `on-close-all` and `can-close-all?`.

- (send *a-group:* `clear`) ⇒ boolean

`for-each-frame`

This method applies a function to each frame in the group. It also remembers the function and applies it to any new frames that are added to the group when they are added.

See also `get-frames`.

- (send *a-group*: for-each-frame *f*) ⇒ void
f: ((instance `frame:basic<%>`) -> void)
 Applies *f* to each frame in the group

`frame-label-changed`

This method is called by frames constructed with `frame:basic-mixin` when their titles change.

- (send *a-group*: frame-label-changed *frame*) ⇒ void
frame: (implements `frame:basic<%>`)
 Updates the windows menu of each frame in the group.

`frame-shown/hidden`

This method is called by instances of `frame:basic%` to notify the frame group that a frame's visibility is changed.

- (send *a-group*: frame-shown/hidden) ⇒ void
 Updates the Windows menus of all of the frames in the frame group.

`get-active-frame`

Returns the frame with the keyboard focus or the first frame in the group.

- (send *a-group*: get-active-frame) ⇒ (implements `frame:basic<%>`)

`get-frames`

Returns the frames in the group.

- (send *a-group*: get-frames) ⇒ (list-of (instance `frame:basic<%>`))

`get-mdi-parent`

The result of this method must be used as the parent frame for each frame in the group.

- (send *a-group*: get-mdi-parent) ⇒ (union #f (instance `frame%`))

get-open-here-frame

Returns the currently saved frame to load new files into.

```
- (send a-group: get-open-here-frame) ⇒ (union #f (is-a?/c frame:editor<%>))
```

insert-frame

Inserts a frame into the group.

```
- (send a-group: insert-frame frame) ⇒ void
  frame: (implements frame:basic<%>)
```

locate-file

Returns the frame that is editing or viewing a particular file.

```
- (send a-group: locate-file) ⇒ (union #f (implements frame:basic<%>))
```

on-close-all

Call this method to close all of the frames in the group. The function `can-close-all?` must have been called just before this function and it must have returned `#t`.

```
- (send a-group: on-close-all) ⇒ void
  Calls the on-close method and the show method (with #f as argument) on each frame in the group.
```

remove-frame

Removes a frame from the group.

```
- (send a-group: remove-frame frame) ⇒ void
  frame: (implements frame:basic<%>)
```

set-active-frame

Sets the active frame in the group. This method is called by `on-focus`.

```
- (send a-group: set-active-frame frame) ⇒ void  
  frame: (implements frame:basic<*>)
```

set-open-here-frame

Sets the frame to load new files into. See also `frame:open-here<*>`.

```
- (send a-group: set-open-here-frame frame) ⇒ void  
  frame: (is-a?/c frame:editor%)
```

14. Handler

14.0.0.1 OPENING A FILE AND SELECTING A FORMAT HANDLER

The function `handler:edit-file` takes a filename and dispatches it to an appropriate *format handler*. A format handler takes a filename and opens a frame for the user to view or edit the file. If no handler is found for a particular format, then the file is opened as a raw text file, in a `frame:text%` object.

The function `handler:open-file` lets the user select a filename using `finder:get-file`, and then passes the name to `handler:edit-file`.

15. Icon

16. Keymap

- `keymap:aug-keymap%`
- `keymap:aug-keymap<%>`
- `keymap:aug-keymap-mixin`

16.1 `keymap:aug-keymap<%>`

Extends: (class->interface `keymap%`)

This keymap overrides some of the built in `keymap%` methods to be able to extract the keybindings from the keymap.

`get-chained-keymaps`

- (send *a-keymap:aug-keymap* `get-chained-keymaps`) ⇒ (listof (instance `keymap%`))
Returns the list of keymaps that are chained to this one.

`get-map-function-table`

- (send *a-keymap:aug-keymap* `get-map-function-table`) ⇒ `hash-table`
Returns a hash-table that maps symbols naming key sequences to the names of the keymap functions the are bound to.

`get-map-function-table/ht`

- (send *a-keymap:aug-keymap* `get-map-function-table/ht` *ht*) ⇒ `hash-table`
ht : `hash-table`

This is a helper function for `get-map-function-table` that returns the same result, except it accepts a hash-table that it inserts the bindings into. It does not replace any bindings already in *ht*.

16.2 keymap:aug-keymap-mixin

Domain: (class->interface `keymap%`)

Implements: `keymap:aug-keymap<%>`

- `init args:`

Creates an empty keymap.

`chain-to-keymap`

Chains a keymap off this one.

Multiple keymaps can be chained off one keymap using `chain-to-keymap`. When keymaps are chained off a main keymap, events not handled by the main keymap are passed to the chained keymaps until some chained keymap handles the events. Keymaps can be chained together in an arbitrary acyclic graph.

Keymap chaining is useful because multiple-event sequences are handled correctly for chained groups. Without chaining, a sequence of events can produce state in a keymap that must be reset when a callback is invoked in one of the keymaps. This state can be manually cleared with `break-sequence`, though calling the `break-sequence` method also invokes the handler installed by `set-break-sequence-callback`.

```
- (send a-keymap:aug-keymap-mixin chain-to-keymap next prefix? void
    next: (instance keymap%)
    prefix?: boolean)
```

Keeps a list of the keymaps chained to this one.

`map-function`

Maps an input state to the name of an event handler.

```
- (send a-keymap:aug-keymap-mixin map-function key-name function-name void
    key-name: string
    function-name: string)
```

Keeps a separate record of the key names and functions that they are bound to in this keymap.

`remove-chained-keymap`

Unchains a keymap from this keymap.

- (send *a-keymap:aug-keymap-mixin* remove-chained-keymap *keymap* void
keymap: `keymap%` object

Keeps the list of the keymaps chained to this one up to date.

16.3 `keymap:aug-keymap% = (keymap:aug-keymap-mixin keymap%)`

`keymap:aug-keymap% = (keymap:aug-keymap-mixin keymap%)`

- (new `keymap:aug-keymap%`) ⇒ `keymap:aug-keymap%` object

Passes all arguments to `super-init`.

17. Main

18. Menu

- `menu:can-restore-checkable-menu-item%`
- `menu:can-restore-menu-item%`
- `menu:can-restore-underscore-menu%`
- `menu:can-restore-underscore<%>`
- `menu:can-restore<%>`
- `menu:can-restore-mixin`
- `menu:can-restore-underscore-mixin`

18.1 `menu:can-restore<%>`

Extends: `selectable-menu-item<%>`

Classes created with this mixin remember their keybindings so the keybindings can be removed and then restored.

`restore-keybinding`

Sets the keyboard shortcut to the setting it had when the class was created.

```
- (send a-menu:can-restore restore-keybinding) ⇒ void
```

18.2 `menu:can-restore-mixin`

Domain: `selectable-menu-item<%>`

Implements: `selectable-menu-item<%>`

Implements: `menu:can-restore<%>`

18.3 menu:can-restore-underscore<%>

Extends: `labelled-menu-item<%>`

These menus can save and restore the underscores (indicated via the & characters in the original labels) in their labels.

If the preference 'framework:menu-bindings is \#f, calls `erase-underscores` during initialization.

`erase-underscores`

Erases the underscores in the label of this menu, but remembers them so they can be restores with `restore-underscores`.

- (send *a-menu:can-restore-underscore* erase-underscores) ⇒ void

`restore-underscores`

Restores underscores in the menu's label to their original state.

- (send *a-menu:can-restore-underscore* restore-underscores) ⇒ void

18.4 menu:can-restore-underscore-mixin

Domain: `labelled-menu-item<%>`

Implements: `menu:can-restore-underscore<%>`

Implements: `labelled-menu-item<%>`

18.5 menu:can-restore-menu-item% = (menu:can-restore-mixin menu-item%)

`menu:can-restore-menu-item% = (menu:can-restore-mixin menu-item%)`

- (new menu:can-restore-menu-item% (label _) (parent _) (callback _) [(shortcut _)] [(help-string _)] [(demand-callback _)]) ⇒ menu:can-restore-menu-item% object
label : string (up to 200 characters)

parent : menu% or popup-menu% object
callback : procedure of two arguments: a menu-item% object and a control-event% object
shortcut = #f : character or #f
help-string = #f : string (up to 200 characters) or #f
demand-callback = void : procedure of one argument: a menu-item% object

Passes all arguments to super-init.

18.6 menu:can-restore-checkable-menu-item% = (menu:can-restore-mixin checkable-menu-item%)

menu:can-restore-checkable-menu-item% = (menu:can-restore-mixin checkable-menu-item%)

- (new menu:can-restore-checkable-menu-item% (label _) (parent _) (callback _)
[(shortcut _)] [(help-string _)] [(demand-callback _)] [(checked _)]) ⇒ menu:can-restore-
object
label : string (up to 200 characters)
parent : menu% or popup-menu% object
callback : procedure of two arguments: a menu-item% object and a control-event% object
shortcut = #f : character or #f
help-string = #f : string (up to 200 characters) or #f
demand-callback = void : procedure of one argument: a checkable-menu-item% object
checked = #f : boolean

Passes all arguments to super-init.

18.7 menu:can-restore-underscore-menu% = (menu:can-restore-underscore-mixin menu%)

menu:can-restore-underscore-menu% = (menu:can-restore-underscore-mixin menu%)

- (new menu:can-restore-underscore-menu% (label _) (parent _) [(help-string _)]
[(demand-callback _)]) ⇒ menu:can-restore-underscore-menu% object
label : string (up to 200 characters)
parent : menu%, popup-menu%, or menu-bar% object
help-string = #f : string (up to 200 characters) or #f
demand-callback = void : procedure of one argument: a menu% object

Passes all arguments to super-init.

19. Mode

19.1 mode:host-text<%>

get-surrogate

- (send *a-mode:host-text* get-surrogate) ⇒ object

Returns the last value passed to `set-surrogate`

set-surrogate

- (send *a-mode:host-text* set-surrogate *surrogate*) ⇒ void
surrogate: (union #f surrogate-object)

Sets the current surrogate object to *surrogate*. If a surrogate has been set before, this method calls that object's `on-disable-surrogate` method. If *surrogate* is not #f, calls its `on-enable-surrogate` method.

19.2 mode:surrogate-text<%>

after-change-style

- (send *a-mode:surrogate-text* after-change-style *orig super-call start len*) ⇒
??
orig: ???
super-call: (??? -i ???)
start: ??
len: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-delete

- (send *a-mode:surrogate-text* after-delete *orig super-call start len*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)
start: ??
len: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-edit-sequence

- (send *a-mode:surrogate-text* after-edit-sequence *orig super-call*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)

Calls *super-call* with the other arguments to this method (except *orig*).

after-insert

- (send *a-mode:surrogate-text* after-insert *orig super-call start len*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)
start: ??
len: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-load-file

- (send *a-mode:surrogate-text* after-load-file *orig super-call success?*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)
success?: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-save-file

- (send *a-mode:surrogate-text* after-save-file *orig super-call success?*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)
success?: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-set-position

```
- (send a-mode:surrogate-text after-set-position orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -i ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

after-set-size-constraint

```
- (send a-mode:surrogate-text after-set-size-constraint orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -i ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-change-style?

```
- (send a-mode:surrogate-text can-change-style? orig super-call start len) ⇒
  ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-delete?

```
- (send a-mode:surrogate-text can-delete? orig super-call start len) ⇒ ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-do-edit-operation? (*augmentable only*)

```
- (send a-mode:surrogate-text can-do-edit-operation? orig super-call op) ⇒
  ???
  orig: ???
  super-call: (??? -i ???)
  op: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

```
- (send a-mode:surrogate-text can-do-edit-operation? orig super-call op recursive?)
⇒ ???
  orig: ???
  super-call: (??? -i ???)
  op: ??
  recursive?: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-insert?

```
- (send a-mode:surrogate-text can-insert? orig super-call start len) ⇒ ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-load-file?

```
- (send a-mode:surrogate-text can-load-file? orig super-call filename format)
⇒ ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  format: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-save-file?

```
- (send a-mode:surrogate-text can-save-file? orig super-call filename format)
⇒ ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  format: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-set-size-constraint?

- (send *a-mode:surrogate-text* can-set-size-constraint? *orig* *super-call*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)

Calls *super-call* with the other arguments to this method (except *orig*).

on-change

- (send *a-mode:surrogate-text* on-change *orig* *super-call*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)

Calls *super-call* with the other arguments to this method (except *orig*).

on-change-style

- (send *a-mode:surrogate-text* on-change-style *orig* *super-call* *start* *len*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)
start: ??
len: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-char (*augmentable only*)

- (send *a-mode:surrogate-text* on-char *orig* *super-call* *event*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)
event: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-default-char (*augmentable only*)

- (send *a-mode:surrogate-text* on-default-char *orig* *super-call* *event*) ⇒ ???
orig: ???
super-call: (??? -*i* ???)
event: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-default-event (*augmentable only*)

```
- (send a-mode:surrogate-text on-default-event orig super-call event) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
  event: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-delete

```
- (send a-mode:surrogate-text on-delete orig super-call start len) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-disable-surrogate

Called when this surrogate is removed from *object*. See also [set-surrogate](#).

```
- (send a-mode:surrogate-text on-disable-surrogate object) ⇒ void
  object: object
```

on-display-size

```
- (send a-mode:surrogate-text on-display-size orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-edit-sequence

```
- (send a-mode:surrogate-text on-edit-sequence orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-enable-surrogate

Called when the surrogate is enabled, with the object it is enabled in. See also [set-surrogate](#).

- (send *a-mode:surrogate-text* on-enable-surrogate *object*) ⇒ void
object: object

on-event (*augmentable only*)

- (send *a-mode:surrogate-text* on-event *orig super-call event*) ⇒ ???
orig: ???
super-call: (??? -i ???)
event: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-focus (*augmentable only*)

- (send *a-mode:surrogate-text* on-focus *orig super-call on?*) ⇒ ???
orig: ???
super-call: (??? -i ???)
on?: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-insert

- (send *a-mode:surrogate-text* on-insert *orig super-call start len*) ⇒ ???
orig: ???
super-call: (??? -i ???)
start: ??
len: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-load-file

- (send *a-mode:surrogate-text* on-load-file *orig super-call filename format*) ⇒
 ???
orig: ???
super-call: (??? -i ???)

```
filename: ??
format: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-local-char (*augmentable only*)

```
- (send a-mode:surrogate-text on-local-char orig super-call event) => ???
  orig: ???
  super-call: (??? -i ???)
  event: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-local-event (*augmentable only*)

```
- (send a-mode:surrogate-text on-local-event orig super-call event) => ???
  orig: ???
  super-call: (??? -i ???)
  event: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-box (*augmentable only*)

```
- (send a-mode:surrogate-text on-new-box orig super-call type) => ???
  orig: ???
  super-call: (??? -i ???)
  type: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-image-snip (*augmentable only*)

```
- (send a-mode:surrogate-text on-new-image-snip orig super-call filename kind
  relative-path? inline?) => ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  kind: ??
  relative-path?: ??
  inline?: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-string-snip (*augmentable only*)

```
- (send a-mode:surrogate-text on-new-string-snip orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-tab-snip (*augmentable only*)

```
- (send a-mode:surrogate-text on-new-tab-snip orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-paint (*augmentable only*)

```
- (send a-mode:surrogate-text on-paint orig super-call before? dc left top right
  bottom dx dy draw-caret) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
  before?: ??
  dc: ??
  left: ??
  top: ??
  right: ??
  bottom: ??
  dx: ??
  dy: ??
  draw-caret: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-save-file

```
- (send a-mode:surrogate-text on-save-file orig super-call filename format) ⇒
  ???
  orig: ???
  super-call: (??? -¿ ???)
  filename: ??
  format: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-set-size-constraint

- (send *a-mode:surrogate-text* on-set-size-constraint *orig* *super-call*) ⇒ ???
orig: ???
super-call: (??? -i ???)

Calls *super-call* with the other arguments to this method (except *orig*).

on-snip-modified

- (send *a-mode:surrogate-text* on-snip-modified *orig* *super-call* *snip* *modified?*) ⇒ ???
orig: ???
super-call: (??? -i ???)
snip: ??
modified?: ??

Calls *super-call* with the other arguments to this method (except *orig*).

19.3 mode:host-text-mixin

Domain: (class->interface `text%`)

Implements: `mode:host-text<%>`

- init args: [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]
line-spacing = 1.0: non-negative real number
tab-stops = null: list of real numbers
auto-wrap = #f: boolean

The *line-spacing* argument sets the additional amount of space (in DC units) inserted between each line in the editor when the editor is displayed. This spacing is included in the reported height of each line.

See `set-tabs` for information about *tabstops*.

If *auto-wrap* is true, then auto-wrapping is enabled via `auto-wrap`.

A new `keymap%` object is created for the new editor. See also `get-keymap` and `set-keymap`.

A new `style-list%` object is created for the new editor. See also `get-style-list` and `set-style-list`.

after-change-style (*augments, and augmentable only*)

Called after the style is changed for a given range (and after the `display` is refreshed; use `on-change-style` and `begin-edit-sequence` to avoid extra refreshes when `after-change-style` modifies the editor).

See also [can-change-style?](#) and [on-edit-sequence](#).

No internals locks are set when this method is called.

```
- (send a-mode:host-text-mixin after-change-style orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`after-delete` (*augments, and augmentable only*)

Called after a given range is deleted from the editor (and after the `display` is refreshed; use [on-delete](#) and [begin-edit-sequence](#) to avoid extra refreshes when `after-delete` modifies the editor).

See also [can-delete?](#) and [on-edit-sequence](#).

No internals locks are set when this method is called.

```
- (send a-mode:host-text-mixin after-delete orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`after-edit-sequence` (*augments, and augmentable only*)

Called after a top-level edit sequence completes (involving unnested [begin-edit-sequence](#) and [end-edit-sequence](#)).

See also [on-edit-sequence](#).

```
- (send a-mode:host-text-mixin after-edit-sequence orig super-call ???
  orig: ???
  super-call: (??? -i ???)
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`after-insert` (*augments, and augmentable only*)

Called after `items` are inserted into the editor (and after the `display` is refreshed; use `on-insert` and `begin-edit-sequence` to avoid extra refreshes when `after-insert` modifies the editor).

See also `can-insert?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```
- (send a-mode:host-text-mixin after-insert orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`after-load-file` (*augments, and augmentable only*)

Called just after the editor is loaded from a file.

The argument to the method originally specified whether the save was successful, but failures now trigger exceptions such that the method is not even called. Consequently, the argument is always `#t`.

See also `can-load-file?` and `on-load-file`.

```
- (send a-mode:host-text-mixin after-load-file orig super-call success? ???
  orig: ???
  super-call: (??? -i ???)
  success?: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`after-save-file` (*augments, and augmentable only*)

Called just after the editor is saved to a file.

The argument to the method originally specified whether the save was successful, but failures now trigger exceptions such that the method is not even called. Consequently, the argument is always `#t`.

See also `can-save-file?` and `on-save-file`.

```
- (send a-mode:host-text-mixin after-save-file orig super-call success? ???
  orig: ???
  super-call: (??? -i ???)
  success?: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`after-set-position` (*augments, and augmentable only*)

Called after the start and end `position` have been moved (but not when the `position` is moved due to inserts or deletes).

See also `on-edit-sequence`.

```
- (send a-mode:host-text-mixin after-set-position orig super-call ???
  orig: ???
  super-call: (??? -i ???)
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`after-set-size-constraint` (*augments, and augmentable only*)

Called after the editor's maximum or minimum height or width is changed (and after the `display` is refreshed; use `on-set-size-constraint` and `begin-edit-sequence` to avoid extra refreshes when `after-set-size-constraint` modifies the editor).

(This callback method is provided because setting an editor's maximum width may cause lines to be re-flowed with soft carriage returns.)

See also `can-set-size-constraint?` and `on-edit-sequence`.

```
- (send a-mode:host-text-mixin after-set-size-constraint orig super-call ???
  orig: ???
  super-call: (??? -i ???)
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`can-change-style?` (*augments, and augmentable only*)

Called before the style is changed in a given range of the editor. If the return value is `#f`, then the style change will be aborted.

The editor is internally locked for writing during a call to this method (see also "Locks" (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-change-style` to modify the editor, if necessary.

See also `on-change-style`, `after-change-style`, and `on-edit-sequence`.

```
- (send a-mode:host-text-mixin can-change-style? orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`can-delete?` (*augments, and augmentable only*)

Called before a range is deleted from the editor. If the return value is `#f`, then the delete will be aborted.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-delete` to modify the editor, if necessary.

See also `on-delete`, `after-delete`, and `on-edit-sequence`.

```
- (send a-mode:host-text-mixin can-delete? orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`can-do-edit-operation?`

Checks whether a generic edit command would succeed for the editor. This check is especially useful for enabling and disabling menus on demand.

```
- (send a-mode:host-text-mixin can-do-edit-operation? orig super-call op ???
  orig: ???
  super-call: (??? -i ???)
  op: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

```
- (send a-mode:host-text-mixin can-do-edit-operation? orig super-call op recursive?
  ???
  orig: ???
  super-call: (??? -i ???)
  op: ??
  recursive?: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`can-insert?` (*augments, and augmentable only*)

Called before `items` are inserted into the editor. If the return value is `#f`, then the insert will be aborted.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-insert` to modify the editor, if necessary.

See also `on-insert`, `after-insert`, and `on-edit-sequence`.

```
- (send a-mode:host-text-mixin can-insert? orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`can-load-file?` (*augments, and augmentable only*)

Called just before the editor is loaded from a file. If the return value is `#f`, the file is not loaded. See also `on-load-file` and `after-load-file`.

```
- (send a-mode:host-text-mixin can-load-file? orig super-call filename format
  ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  format: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`can-save-file?` (*augments, and augmentable only*)

Called just before the editor is saved to a file. If the return value is `#f`, the file is not saved. See also `on-save-file` and `after-save-file`.

```
- (send a-mode:host-text-mixin can-save-file? orig super-call filename format
  ???
```

```

orig: ???
super-call: (??? -i ???)
filename: ??
format: ??

```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`can-set-size-constraint?` (*augments, and augmentable only*)

Called before the editor's maximum or minimum height or width is changed. If the return value is `#f`, then the change will be aborted.

(This callback method is provided because setting an editor's maximum width may cause lines to be re-flowed with soft carriage returns.)

See also `on-set-size-constraint`, `after-set-size-constraint`, and `on-edit-sequence`.

```

- (send a-mode:host-text-mixin can-set-size-constraint? orig super-call ???
  orig: ???
  super-call: (??? -i ???)

```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-change` (*augments, and augmentable only*)

Called whenever any change is made to the editor that affects the way the editor is drawn or the values reported for the `location`/size of some snip in the editor. The `on-change` method is called just before the editor calls its administrator's `needs-update` method to refresh the editor's `display`, and it is also called just before and after printing an editor.

The editor is locked for writing and reflowing during the call to `on-change`.

```

- (send a-mode:host-text-mixin on-change orig super-call ???
  orig: ???
  super-call: (??? -i ???)

```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-change-style` (*augments, and augmentable only*)

Called before the style is changed in a given range of the editor, after `can-change-style?` is called to verify that the change is ok. The `after-change-style` method is guaranteed to be called after the change has completed.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-change-style` to modify the editor, if necessary.

See also `on-edit-sequence`.

```
- (send a-mode:host-text-mixin on-change-style orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-char`

Handles keyboard input to the editor.

Consider overriding `on-local-char` or `on-default-char` instead of this method.

```
- (send a-mode:host-text-mixin on-char orig super-call event ???
  orig: ???
  super-call: (??? -i ???)
  event: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-default-char`

Called by `on-local-char` when the event is *not* handled by a caret-owning snip or by the keymap.

```
- (send a-mode:host-text-mixin on-default-char orig super-call event ???
  orig: ???
  super-call: (??? -i ???)
  event: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-default-event`

Called by `on-local-event` when the event is *not* handled by a caret-owning snip or by the keymap.

```
- (send a-mode:host-text-mixin on-default-event orig super-call event ???
  orig: ???
  super-call: (??? -i ???)
  event: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-delete` (*augments, and augmentable only*)

Called before a range is deleted from the editor, after `can-delete?` is called to verify that the deletion is ok. The `after-delete` method is guaranteed to be called after the delete has completed.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-delete` to modify the editor, if necessary.

See also `on-edit-sequence`.

```
- (send a-mode:host-text-mixin on-delete orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-display-size` (*augments, and augmentable only*)

This method is called by the editor’s `display` whenever the display’s size (as reported by `get-view-size`) changes, but it is called indirectly through `on-display-size-when-ready`.

```
- (send a-mode:host-text-mixin on-display-size orig super-call ???
  orig: ???
  super-call: (??? -i ???)
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-edit-sequence` (*augments, and augmentable only*)

Called just after a top-level (i.e., unnested) edit sequence starts.

During an edit sequence, all callbacks methods are invoked normally, but it may be appropriate for these callbacks to delay computation during an edit sequence. The callbacks must manage this delay manually.

Thus, when overriding other callback methods, such as `on-insert in text%`, `on-insert in pasteboard%`, `after-insert in text%`, or `after-insert in pasteboard%`, consider overriding `on-edit-sequence` and `after-edit-sequence` as well.

“Top-level edit sequence” refers to an outermost pair of `begin-edit-sequence` and `end-edit-sequence` calls. The embedding of an editor within another editor does not affect the timing of calls to `on-edit-sequence`, even if the embedding editor is in an edit sequence.

Pairings of `on-edit-sequence` and `after-edit-sequence` can be nested if an `after-edit-sequence` starts a new edit sequence, since `after-edit-sequence` is called after an edit sequence ends. However, `on-edit-sequence` can never start a new top-level edit sequence (except through an unpaired `end-edit-sequence`), because it is called after a top-level edit sequence starts.

```
- (send a-mode:host-text-mixin on-edit-sequence orig super-call ???
    orig: ???
    super-call: (??? -i ???)
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-event`

Handles mouse input to the editor. The event’s x and y coordinates are in the `display`’s co-ordinate system; use the administrator’s `get-dc` method to obtain translation arguments (or use `dc-location-to-editor-location`).

Consider overriding `on-local-event` or `on-default-event` instead of this method.

```
- (send a-mode:host-text-mixin on-event orig super-call event ???
    orig: ???
    super-call: (??? -i ???)
    event: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-focus`

Called when the keyboard focus changes into or out of this editor (and not to/from a snip within the editor) with `#t` if the focus is being turned on, `#f` otherwise.

```
- (send a-mode:host-text-mixin on-focus orig super-call on? ???
    orig: ???
    super-call: (??? -i ???)
    on?: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

on-insert (*augments, and augmentable only*)

Called before **items** are inserted into the editor, after **can-insert?** is called to verify that the insertion is ok. The **after-insert** method is guaranteed to be called after the insert has completed.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use **after-insert** to modify the editor, if necessary.

See also **on-edit-sequence**.

```
- (send a-mode:host-text-mixin on-insert orig super-call start len ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus **this** and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

on-load-file (*augments, and augmentable only*)

Called just before the editor is loaded from a file, after calling **can-load-file?** to verify that the load is allowed. See also **after-load-file**.

```
- (send a-mode:host-text-mixin on-load-file orig super-call filename format ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  format: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus **this** and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

on-local-char

Called by **on-char** when the event is *not* handled by a caret-owning snip.

Consider overriding **on-default-char** instead of this method.

```
- (send a-mode:host-text-mixin on-local-char orig super-call event ???
  orig: ???
  super-call: (??? -i ???)
  event: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus **this** and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-local-event`

Called by `on-event` when the event is *not* handled by a caret-owning snip.

Consider overriding `on-default-event` instead of this method.

```
- (send a-mode:host-text-mixin on-local-event orig super-call event ???
  orig: ???
  super-call: (??? -i ???)
  event: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-new-box`

Creates and returns a new snip for an embedded editor. This method is called by `insert-box`.

```
- (send a-mode:host-text-mixin on-new-box orig super-call type ???
  orig: ???
  super-call: (??? -i ???)
  type: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-new-image-snip`

Creates and returns a new instance of `image-snip%` for `insert-image`.

```
- (send a-mode:host-text-mixin on-new-image-snip orig super-call filename kind
  relative-path? inline? ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  kind: ??
  relative-path?: ??
  inline?: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

on-new-string-snip

Called by `insert` when a string or character is inserted into the editor, this method creates and returns a new instance of `string-snip%` to store inserted text. The returned string snip is empty (i.e., its `count` is zero).

```
- (send a-mode:host-text-mixin on-new-string-snip orig super-call ???
    orig: ???
    super-call: (??? -i ???))
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

on-new-tab-snip

Creates and returns a new instance of `tab-snip%` to store an inserted tab. The returned tab snip is empty (i.e., its `count` is zero).

```
- (send a-mode:host-text-mixin on-new-tab-snip orig super-call ???
    orig: ???
    super-call: (??? -i ???))
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

on-paint

Provides a way to add arbitrary graphics to an editor's `display`. This method is called just before and just after every painting of the editor.

The `on-paint` method, together with the snips' `draw` methods, must be able to draw the entire state of an editor. Never paint directly into an editor's `display` canvas except from within `on-paint` or `draw`. Instead, put all extra drawing code within `on-paint` and call `invalidate-bitmap-cache` when part of the `display` needs to be repainted.

If an `on-paint` method uses cached `location` information, then the cached information should be recomputed in response to a call of `invalidate-bitmap-cache`.

The `on-paint` method must not make any assumptions about the state of the drawing context (e.g., the current pen), except that the clipping region is already set to something appropriate. Before `on-paint` returns, it must restore any drawing context settings that it changes.

The editor is internally locked for writing and reflowing during a call to this method (see also "Locks" (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-mode:host-text-mixin on-paint orig super-call before? dc left top right
    bottom dx dy draw-caret ???)
```

```

orig: ???
super-call: (??? -i ???)
before?: ??
dc: ??
left: ??
top: ??
right: ??
bottom: ??
dx: ??
dy: ??
draw-caret: ??

```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-save-file` (*augments, and augmentable only*)

Called just before the editor is saved to a file, after calling `can-save-file?` to verify that the save is allowed. See also `after-save-file`.

```

- (send a-mode:host-text-mixin on-save-file orig super-call filename format ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  format: ??

```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

`on-set-size-constraint` (*augments, and augmentable only*)

Called before the editor's maximum or minimum height or width is changed, after `can-set-size-constraint?` is called to verify that the change is ok. The `after-set-size-constraint` method is guaranteed to be called after the change has completed.

(This callback method is provided because setting an editor's maximum width may cause lines to be re-flowed with soft carriage returns.)

See also `on-edit-sequence`.

```

- (send a-mode:host-text-mixin on-set-size-constraint orig super-call ???
  orig: ???
  super-call: (??? -i ???)

```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

on-snip-modified (*augments, and augmentable only*)

This method is called whenever a snip within the editor reports that it has been modified (by calling its administrator's `modified` method). The method arguments are the snip that reported a modification-state change, and the snip's new modification state.

See also `set-modified`.

```
- (send a-mode:host-text-mixin on-snip-modified orig super-call snip modified?
  ???
  orig: ???
  super-call: (??? -i ???)
  snip: ??
  modified?: ??
```

If a surrogate is set, calls the surrogate with the arguments it was passed, plus `this` and a function that calls the super method.

if a surrogate is not set, calls the super method and returns that result.

19.4 mode:surrogate-text%

Implements: `mode:surrogate-text<%>`

```
- (new mode:surrogate-text% ) => mode:surrogate-text% object
```

after-change-style (*augmentable only*)

```
- (send a-mode:surrogate-text after-change-style orig super-call start len) =>
  ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls `super-call` with the other arguments to this method (except `orig`).

after-delete (*augmentable only*)

```
- (send a-mode:surrogate-text after-delete orig super-call start len) => ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

after-edit-sequence (*augmentable only*)

- (send *a-mode:surrogate-text* after-edit-sequence *orig* *super-call*) ⇒ ???
orig: ???
super-call: (??? -i ???)

Calls *super-call* with the other arguments to this method (except *orig*).

after-insert (*augmentable only*)

- (send *a-mode:surrogate-text* after-insert *orig* *super-call* *start* *len*) ⇒ ???
orig: ???
super-call: (??? -i ???)
start: ??
len: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-load-file (*augmentable only*)

- (send *a-mode:surrogate-text* after-load-file *orig* *super-call* *success?*) ⇒ ???
orig: ???
super-call: (??? -i ???)
success?: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-save-file (*augmentable only*)

- (send *a-mode:surrogate-text* after-save-file *orig* *super-call* *success?*) ⇒ ???
orig: ???
super-call: (??? -i ???)
success?: ??

Calls *super-call* with the other arguments to this method (except *orig*).

after-set-position (*augmentable only*)

- (send *a-mode:surrogate-text* after-set-position *orig* *super-call*) ⇒ ???
orig: ???
super-call: (??? -i ???)

Calls *super-call* with the other arguments to this method (except *orig*).

after-set-size-constraint (*augmentable only*)

```
- (send a-mode:surrogate-text after-set-size-constraint orig super-call) => ???
  orig: ???
  super-call: (??? -i ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-change-style? (*augmentable only*)

```
- (send a-mode:surrogate-text can-change-style? orig super-call start len) =>
  ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-delete? (*augmentable only*)

```
- (send a-mode:surrogate-text can-delete? orig super-call start len) => ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

can-do-edit-operation?

```
- (send a-mode:surrogate-text can-do-edit-operation? orig super-call op) =>
  ???
  orig: ???
  super-call: (??? -i ???)
  op: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

```
- (send a-mode:surrogate-text can-do-edit-operation? orig super-call op recursive?)
  => ???
  orig: ???
  super-call: (??? -i ???)
```

```

op: ??
recursive?: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

can-insert? (*augmentable only*)

```

- (send a-mode:surrogate-text can-insert? orig super-call start len) => ???
  orig: ???
  super-call: (??? -¿ ???)
  start: ??
  len: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

can-load-file? (*augmentable only*)

```

- (send a-mode:surrogate-text can-load-file? orig super-call filename format)
  => ???
  orig: ???
  super-call: (??? -¿ ???)
  filename: ??
  format: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

can-save-file? (*augmentable only*)

```

- (send a-mode:surrogate-text can-save-file? orig super-call filename format)
  => ???
  orig: ???
  super-call: (??? -¿ ???)
  filename: ??
  format: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

can-set-size-constraint? (*augmentable only*)

```

- (send a-mode:surrogate-text can-set-size-constraint? orig super-call) => ???
  orig: ???
  super-call: (??? -¿ ???)

```

Calls *super-call* with the other arguments to this method (except *orig*).

on-change (*augmentable only*)

- (send *a-mode:surrogate-text* on-change *orig* *super-call*) ⇒ ???
 orig: ???
 super-call: (??? -*i* ???)

Calls *super-call* with the other arguments to this method (except *orig*).

on-change-style (*augmentable only*)

- (send *a-mode:surrogate-text* on-change-style *orig* *super-call* *start* *len*) ⇒ ???
 orig: ???
 super-call: (??? -*i* ???)
 start: ??
 len: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-char

- (send *a-mode:surrogate-text* on-char *orig* *super-call* *event*) ⇒ ???
 orig: ???
 super-call: (??? -*i* ???)
 event: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-default-char

- (send *a-mode:surrogate-text* on-default-char *orig* *super-call* *event*) ⇒ ???
 orig: ???
 super-call: (??? -*i* ???)
 event: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-default-event

- (send *a-mode:surrogate-text* on-default-event *orig* *super-call* *event*) ⇒ ???
 orig: ???
 super-call: (??? -*i* ???)
 event: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-delete (*augmentable only*)

```
- (send a-mode:surrogate-text on-delete orig super-call start len) ⇒ ???
  orig: ???
  super-call: (??? -i ???)
  start: ??
  len: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-disable-surrogate

Called when this surrogate is removed from *object*. See also [set-surrogate](#).

```
- (send a-mode:surrogate-text on-disable-surrogate object) ⇒ void
  object: object
```

Does nothing.

on-display-size (*augmentable only*)

```
- (send a-mode:surrogate-text on-display-size orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -i ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-edit-sequence (*augmentable only*)

```
- (send a-mode:surrogate-text on-edit-sequence orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -i ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-enable-surrogate

Called when the surrogate is enabled, with the object it is enabled in. See also [set-surrogate](#).

```
- (send a-mode:surrogate-text on-enable-surrogate object) ⇒ void
  object: object
```

Does nothing.

on-event

- (send *a-mode:surrogate-text* on-event *orig* *super-call* *event*) ⇒ ???
 - orig*: ???
 - super-call*: (??? -i ???)
 - event*: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-focus

- (send *a-mode:surrogate-text* on-focus *orig* *super-call* *on?*) ⇒ ???
 - orig*: ???
 - super-call*: (??? -i ???)
 - on?*: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-insert (*augmentable only*)

- (send *a-mode:surrogate-text* on-insert *orig* *super-call* *start* *len*) ⇒ ???
 - orig*: ???
 - super-call*: (??? -i ???)
 - start*: ??
 - len*: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-load-file (*augmentable only*)

- (send *a-mode:surrogate-text* on-load-file *orig* *super-call* *filename* *format*) ⇒ ???
 - ???
 - orig*: ???
 - super-call*: (??? -i ???)
 - filename*: ??
 - format*: ??

Calls *super-call* with the other arguments to this method (except *orig*).

on-local-char

- (send *a-mode:surrogate-text* on-local-char *orig* *super-call* *event*) ⇒ ???
 - orig*: ???

```

super-call: (??? -i ???)
event: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

on-local-event

```

- (send a-mode:surrogate-text on-local-event orig super-call event) => ???
  orig: ???
  super-call: (??? -i ???)
  event: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-box

```

- (send a-mode:surrogate-text on-new-box orig super-call type) => ???
  orig: ???
  super-call: (??? -i ???)
  type: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-image-snip

```

- (send a-mode:surrogate-text on-new-image-snip orig super-call filename kind
  relative-path? inline?) => ???
  orig: ???
  super-call: (??? -i ???)
  filename: ??
  kind: ??
  relative-path?: ??
  inline?: ??

```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-string-snip

```

- (send a-mode:surrogate-text on-new-string-snip orig super-call) => ???
  orig: ???
  super-call: (??? -i ???)

```

Calls *super-call* with the other arguments to this method (except *orig*).

on-new-tab-snip

```
- (send a-mode:surrogate-text on-new-tab-snip orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-paint

```
- (send a-mode:surrogate-text on-paint orig super-call before? dc left top right
  bottom dx dy draw-caret) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
  before?: ??
  dc: ??
  left: ??
  top: ??
  right: ??
  bottom: ??
  dx: ??
  dy: ??
  draw-caret: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-save-file (*augmentable only*)

```
- (send a-mode:surrogate-text on-save-file orig super-call filename format) ⇒
  ???
  orig: ???
  super-call: (??? -¿ ???)
  filename: ??
  format: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-set-size-constraint (*augmentable only*)

```
- (send a-mode:surrogate-text on-set-size-constraint orig super-call) ⇒ ???
  orig: ???
  super-call: (??? -¿ ???)
```

Calls *super-call* with the other arguments to this method (except *orig*).

on-snip-modified (*augmentable only*)

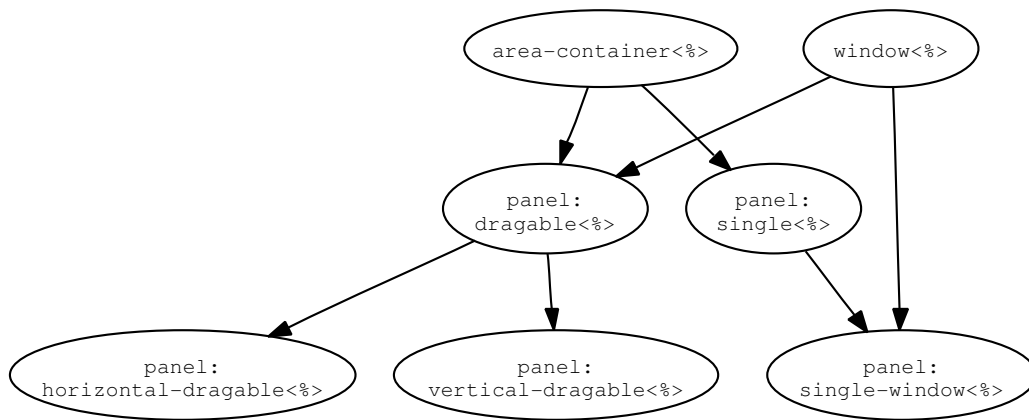
```
- (send a-mode:surrogate-text on-snip-modified orig super-call snip modified?)  
⇒ ???  
  orig: ???  
  super-call: (??? -i ???)  
  snip: ??  
  modified?: ??
```

Calls *super-call* with the other arguments to this method (except *orig*).

20. Panel

This chapter describes the pane mixins, interfaces, and classes.

This is the interface hierarchy:



- `panel:horizontal-draggable%`
- `panel:single-pane%`
- `panel:single%`
- `panel:vertical-draggable%`
- `panel:draggable<%>`
- `panel:horizontal-draggable<%>`
- `panel:single-window<%>`
- `panel:single<%>`
- `panel:vertical-draggable<%>`
- `panel:draggable-mixin`
- `panel:horizontal-draggable-mixin`
- `panel:single-mixin`
- `panel:single-window-mixin`
- `panel:vertical-draggable-mixin`

20.1 panel:single<%>

Extends: `area-container<%>`

See `panel:single-mixin%`.

active-child

- (send *a-panel:single* active-child *child*) ⇒ void
child: (implements `area<%>`)
 Sets the active child to be *child*
- (send *a-panel:single* active-child) ⇒ (implements `area<%>`)
 Returns the current active child.

20.2 panel:single-mixin

Domain: `area-container<%>`

Implements: `panel:single<%>`

Implements: `area-container<%>`

This mixin adds single panel functionality to an implementation of the `area-container<%>` interface.

Single panels place all of the children in the center of the panel, and allow make one child to be visible at a time. The `active-child` method controls which panel is currently active.

The `show` method is used to hide and show the children of a single panel.

after-new-child

This method is called after a new containee area is created with this area as its container. The new child is provided as an argument to the method.

- (send *a-panel:single-mixin* after-new-child *child* void
child: `subarea<%>`
 Hides this child by calling

 (send *child* show #f)

, unless this is the first child in which case it does nothing.

`container-size`

Called to determine the minimum size of a container. See Geometry Management, §2.2 in *PLT MrEd: Graphical Toolbox Manual* for more information.

- (send *a-panel:single-mixin* container-size (values exact-integer exact-integer)
Returns the maximum width of all the children and the maximum height of all of the children.

`place-children`

Called to place the children of a container. See Geometry Management, §2.2 in *PLT MrEd: Graphical Toolbox Manual* for more information.

- (send *a-panel:single-mixin* place-children (listof (list exact-integer exact-integer exact-integer exact-integer))
Returns the positions for single panels and panes.

20.3 `panel:single-window<%>`

Extends: `panel:single<%>`

Extends: `window<%>`

20.4 `panel:single-window-mixin`

Domain: `panel:single<%>`

Domain: `window<%>`

Implements: `panel:single-window<%>`

Implements: `panel:single<%>`

Implements: `window<%>`

20. *Panel*. `panel:single%` = `(panel:single-mixin (panel:single-window-mixin panel%))`

`container-size`

Called to determine the minimum size of a container. See Geometry Management, §2.2 in *PLT MrEd: Graphical Toolbox Manual* for more information.

- (send *a-panel:single-window-mixin* `container-size` *info* (values `exact-integer exact-integer`)
info: (list-of (list `exact-integer exact-integer boolean boolean`))

Factors the border width into the size calculation.

20.5 `panel:single%` = `(panel:single-mixin (panel:single-window-mixin panel%))`

`panel:single%` = `(panel:single-mixin (panel:single-window-mixin panel%))`

- (new `panel:single%` (*parent* `-`) [(*style* `-`)] [(*enabled* `-`)] [(*vert-margin* `-`)] [(*horiz-margin* `-`)] [(*border* `-`)] [(*spacing* `-`)] [(*alignment* `-`)] [(*min-width* `-`)] [(*min-height* `-`)] [(*stretchable-width* `-`)] [(*stretchable-height* `-`)] ⇒ `panel:single%` object
parent: `frame%`, `dialog%`, `panel%`, or `pane%` object
style = `null`: list of symbols in ' (`border` deleted)
enabled = `#t`: `boolean`
vert-margin = 0: `exact integer in [0, 1000]`
horiz-margin = 0: `exact integer in [0, 1000]`
border = 0: `exact integer in [0, 1000]`
spacing = 0: `exact integer in [0, 1000]`
alignment = '(`center center`): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
min-width = 0: `exact integer in [0, 10000]`
min-height = 0: `exact integer in [0, 10000]`
stretchable-width = `#t`: `boolean`
stretchable-height = `#t`: `boolean`

Passes all arguments to `super-init`.

20.6 `panel:single-pane%` = `(panel:single-mixin pane%)`

`panel:single-pane%` = `(panel:single-mixin pane%)`

- (new `panel:single-pane%` (*parent* `-`) [(*vert-margin* `-`)] [(*horiz-margin* `-`)] [(*border* `-`)] [(*spacing* `-`)] [(*alignment* `-`)] [(*min-width* `-`)] [(*min-height* `-`)] [(*stretchable-width* `-`)] [(*stretchable-height* `-`)] ⇒ `panel:single-pane%` object
parent: `frame%`, `dialog%`, `panel%`, or `pane%` object
vert-margin = 0: `exact integer in [0, 1000]`
horiz-margin = 0: `exact integer in [0, 1000]`
border = 0: `exact integer in [0, 1000]`
spacing = 0: `exact integer in [0, 1000]`
alignment = '(`center center`): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
min-width = 0: `exact integer in [0, 10000]`
min-height = 0: `exact integer in [0, 10000]`

```
stretchable-width = #t : boolean
stretchable-height = #t : boolean
```

Passes all arguments to super-init.

20.7 panel:dragable<%>

Extends: `area-container<%>`

Extends: `window<%>`

Classes matching this interface implement a panel where the user can adjust the percentage of the space that each takes up. The user adjusts the size by clicking and dragging the empty space between the children.

`after-percentage-change`

This method is called when the user changes the percentage by dragging the bar between the children, or when a new child is added to the frame, but not when `set-percentages` is called.

Use `get-percentages` to find the current percentages.

```
- (send a-panel:dragable after-percentage-change) ⇒ void
```

`get-percentages`

Return the current percentages of the children.

```
- (send a-panel:dragable get-percentages) ⇒ (listof numbers)
```

`get-vertical?`

This method controls the behavior of the other overridden methods in mixins that implement this interface.

If it returns `#t`, the panel will be vertically aligned and if it returns `#f`, they will be horizontally aligned.

```
- (send a-panel:dragable get-vertical?) ⇒ boolean
```

`set-percentages`

Call this method to set the percentages that each window takes up of the panel.

- (send *a-panel:dragable* set-percentages *new-percentages*) ⇒ void
new-percentages: (listof number)

The argument, *new-percentages* must be a list of numbers that sums to 1. It's length must be equal to the number of children of the panel (see `get-children`) and each percentage must correspond to a number of pixels that is equal to or larger than the minimum width of the child, as reported by `min-width`.

20.8 panel:vertical-dragable<%>

Extends: `panel:dragable<%>`

A panel that implements `panel:vertical-dragable<%>`. It aligns its children vertically.

20.9 panel:horizontal-dragable<%>

Extends: `panel:dragable<%>`

A panel that implements `panel:horizontal-dragable<%>`. It aligns its children horizontally.

20.10 panel:dragable-mixin

Domain: `area-container<%>`

Domain: `window<%>`

Implements: `area-container<%>`

Implements: `window<%>`

Implements: `panel:dragable<%>`

This mixin adds the `panel:dragable<%>` functionality to a `panel%`.

It is not useful unless the `get-vertical?` method is overridden.

`after-new-child`

This method is called after a new containee area is created with this area as its container. The new child is provided as

an argument to the method.

- (send *a-panel:dragable-mixin* after-new-child *child* void
child: (instance-of (implements *area*<%>))

Updates the number of percentages to make sure that it matches the number of children and calls *after-percentage-change*.

container-size

Called to determine the minimum size of a container. See Geometry Management, §2.2 in *PLT MrEd: Graphical Toolbox Manual* for more information.

- (send *a-panel:dragable-mixin* container-size *info* two exact integers in [0, 10000]
info: list of list containing two exact integers in [0, 10000] and two booleans

Computes the minimum size the panel would have to be in order to have the current percentages (see *get-percentages*).

on-subwindow-event

Called when this window or a child window receives a mouse event. The *on-subwindow-event* method of the receiver's top-level window is called first (see *get-top-level-window*); if the return value is #f, the *on-subwindow-event* method is called for the next child in the path to the receiver, and so on. Finally, if the receiver's *on-subwindow-event* method returns #f, the event is passed on to the receiver's normal mouse-handling mechanism.

- (send *a-panel:dragable-mixin* on-subwindow-event *receiver* *event* boolean
receiver: (instanceof *window*<%>)
event: (instanceof *mouse-event*%)

When the cursor is dragging the middle bar around, this method handles the resizing of the two panes.

place-children

Called to place the children of a container. See Geometry Management, §2.2 in *PLT MrEd: Graphical Toolbox Manual* for more information.

- (send *a-panel:dragable-mixin* place-children *info* *w* *h* (list-of (list exact-int exact-int exact-int exact-int))
info: (list-of (list exact-int exact-int))
w: exact-int
h: exact-int

Places the children vertically in the panel, based on the percentages returned from *get-percentages*. Also leaves a little gap between each pair of children.

20.11 panel:vertical-dragable-mixin

Domain: `panel:dragable<%>`

Implements: `panel:vertical-dragable<%>`

Implements: `panel:dragable<%>`

This mixin merely overrides the `get-vertical?` method of the `panel:dragable-mixin` to return `#t`.

`get-vertical?`

This method controls the behavior of the other overridden methods in mixins that implement this interface.

If it returns `#t`, the panel will be vertically aligned and if it returns `#f`, they will be horizontally aligned.

```
- (send a-panel:vertical-dragable-mixin get-vertical? boolean
  Returns #t.
```

20.12 panel:horizontal-dragable-mixin

Domain: `panel:dragable<%>`

Implements: `panel:vertical-dragable<%>`

Implements: `panel:dragable<%>`

This mixin merely overrides the `get-vertical?` method of the `panel:dragable-mixin` to return `#f`.

`get-vertical?`

This method controls the behavior of the other overridden methods in mixins that implement this interface.

If it returns `#t`, the panel will be vertically aligned and if it returns `#f`, they will be horizontally aligned.

```
- (send a-panel:horizontal-dragable-mixin get-vertical? boolean
  Returns #f.
```

20.13 `panel:vertical-dragable%` = (`panel:vertical-dragable-mixin` (`panel:dragable-mixin vertical-panel%`))

`panel:vertical-dragable%` = (`panel:vertical-dragable-mixin` (`panel:dragable-mixin vertical-panel%`))

```
- (new panel:vertical-dragable% (parent _) [(style _)] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)]) => panel:vertical-dragable-object
  parent: frame%, dialog%, panel%, or pane% object
  style = null: list of symbols in ' (border deleted)
  enabled = #t: boolean
  vert-margin = 0: exact integer in [0, 1000]
  horiz-margin = 0: exact integer in [0, 1000]
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(left center): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to `super-init`.

20.14 `panel:horizontal-dragable%` = (`panel:horizontal-dragable-mixin`
(`panel:dragable-mixin horizontal-panel%`))

`panel:horizontal-dragable%` = (`panel:horizontal-dragable-mixin` (`panel:dragable-mixin horizontal-panel%`))

```
- (new panel:horizontal-dragable% (parent _) [(style _)] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)]) => panel:horizontal-dragable-object
  parent: frame%, dialog%, panel%, or pane% object
  style = null: list of symbols in ' (border deleted)
  enabled = #t: boolean
  vert-margin = 0: exact integer in [0, 1000]
  horiz-margin = 0: exact integer in [0, 1000]
  border = 0: exact integer in [0, 1000]
  spacing = 0: exact integer in [0, 1000]
  alignment = '(left center): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0: exact integer in [0, 10000]
  min-height = 0: exact integer in [0, 10000]
  stretchable-width = #t: boolean
  stretchable-height = #t: boolean
```

Passes all arguments to `super-init`.

21. Pasteboard

- `pasteboard:backup-autosave%`
- `pasteboard:basic%`
- `pasteboard:file%`
- `pasteboard:info%`
- `pasteboard:keymap%`
- `pasteboard:standard-style-list%`

21.1 `pasteboard:basic% = (editor:basic-mixin pasteboard%)`

```
pasteboard:basic% = (editor:basic-mixin pasteboard%)
```

- (new `pasteboard:basic%`) ⇒ `pasteboard:basic%` object

Passes all arguments to `super-init`.

21.2 `pasteboard:standard-style-list% = (editor:standard-style-list-mixin pasteboard:basic%)`

```
pasteboard:standard-style-list% = (editor:standard-style-list-mixin pasteboard:basic%)
```

- (new `pasteboard:standard-style-list%`) ⇒ `pasteboard:standard-style-list%` object

Passes all arguments to `super-init`.

21.3 `pasteboard:keymap% = (editor:keymap-mixin pasteboard:standard-style-list%)`

```
pasteboard:keymap% = (editor:keymap-mixin pasteboard:standard-style-list%)
```

- (new `pasteboard:keymap%`) ⇒ `pasteboard:keymap%` object

Passes all arguments to `super-init`.

21.4 `pasteboard:file% = (editor:file-mixin pasteboard:keymap%)`

```
pasteboard:file% = (editor:file-mixin pasteboard:keymap%)
```

- (new `pasteboard:file%`) ⇒ `pasteboard:file%` object

Passes all arguments to `super-init`.

21.5 `pasteboard:backup-autosave% = (editor:backup-autosave-mixin pasteboard:f`

```
pasteboard:backup-autosave% = (editor:backup-autosave-mixin pasteboard:file%)
```

- (new `pasteboard:backup-autosave%`) ⇒ `pasteboard:backup-autosave%` object

Passes all arguments to `super-init`.

21.6 `pasteboard:info% = (editor:info-mixin pasteboard:backup-autosave%)`

```
pasteboard:info% = (editor:info-mixin pasteboard:backup-autosave%)
```

- (new `pasteboard:info%`) ⇒ `pasteboard:info%` object

Passes all arguments to `super-init`.

22. Pathname Utilities

23. Preferences

The framework provides a user preferences manager. It provides facilities for getting, setting, marshalling and unmarshalling the user's preferences as well as utilities to manage a preferences dialog box.

The functions used to manage the preferences, `preferences:get`, `preferences:set`, `preferences:set-default`, `preferences:set-un/marshall`, must be called in a certain order. Overall, the defaults must be set first, the marshalling functions next and then getting and setting is allowed. See the individual functions for the precise constraints.

In addition to the functions `preferences:add-scheme-checkbox-panel`, `preferences:add-warnings-checkbox-p`, `preferences:add-editor-checkbox-panel`, and `preferences:add-font-panel` listed here, `scheme:add-preferences-panel` also adds panels to the preferences dialog.

24. Scheme

- `scheme:sexp-snip%`
- `scheme:text-mode%`
- `scheme:text%`
- `scheme:sexp-snip<%>`
- `scheme:text-mode<%>`
- `scheme:text<%>`
- `scheme:set-mode-mixin`
- `scheme:text-mixin`
- `scheme:text-mode-mixin`

24.1 `scheme:sexp-snip<%>`

```
get-saved-snips
```

This returns the list of snips hidden by the sexp snip.

```
- (send a-scheme:sexp-snip get-saved-snips) ⇒ (listof snip%)
```

24.2 `scheme:sexp-snip%`

Superclass: `snip%`

Implements: `scheme:sexp-snip<%>`

Implements: `readable-snip<%>`

```
- (make-object scheme:sexp-snip%) ⇒ scheme:sexp-snip% object
```

Creates a plain snip of length 1 with the "Basic" style of `the-style-list`.

`copy`

Creates and returns a copy of this snip. The `copy` method is responsible for copying this snip's style (as returned by `get-style`) to the new snip.

```
- (send a-scheme:sexp-snip copy) ⇒ (is-a?/c scheme:sexp-snip%)
```

Returns a copy of this snip that includes the hidden snips.

`draw`

Called (by an editor) to draw the snip.

Before this method is called, the correct font, text color, and pen color will have been set in the drawing context for this snip already. (This is *not* true for `get-extent` or `partial-offset`.) The `draw` method must not make any other assumptions about the state of the drawing context, except that the clipping region is already set to something appropriate. Before `draw` returns, it must restore any drawing context settings that it changes.

See also `on-paint in editor<%>`.

The snip's editor is usually internally locked for writing and reflowing when this method is called (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-scheme:sexp-snip draw dc x y left top right bottom dx dy draw-caret)
⇒ void
  dc:      dc<%> object
  x:      real number
  y:      real number
  left:   real number
  top:    real number
  right:  real number
  bottom: real number
  dx:     real number
  dy:     real number
  draw-caret: symbol in '(no-caret show-inactive-caret show-caret)
```

Draws brackets with a centered ellipses between them.

`get-extent`

Calculates the snip's width, height, descent (amount of height which is drawn below the baseline), space (amount of height which is “filler” space at the top), and horizontal spaces (amount of width which is “filler” space at the left and right).

This method is called by the snip's administrator; it should not be called directly by others. To get the extent of a snip, use `get-snip-location in editor<%>`.

A drawing context is provided for the purpose of finding font sizes, but no drawing should occur. The `get-extent` and `partial-offset` methods must not make any assumptions about the state of the drawing context, except that

it is scaled properly. In particular, the font for the snip’s style is not automatically set in the drawing context before the method is called.¹ If `get-extent` or `partial-offset` changes the drawing context’s setting, it must restore them before returning. However, the methods should not need to change the drawing context; only font settings can affect measurement results from a device context, and `get-text-extent in dc<%>` accepts a `font%` argument for sizing that overrides that device context’s current font.

The snip’s left and top `locations` are provided in editor coordinates. In a text editor, the y-coordinate is the *line*’s top `location`; the snip’s actual top `location` is potentially undetermined until its height is known.

If a snip caches the result size for future replies, it should invalidate its cached size when `size-cache-invalid` is called (especially if the snip’s size depends on any device context properties).

If a snip’s size changes after receiving a call to `get-extent` and before receiving a call to `size-cache-invalid`, then the snip must notify its administrator of the size change, so that the administrator can recompute its derived size information. Notify the administrator of a size change by call its `resized` method.

The snip’s editor is usually internally locked for writing and reflowing when this method is called (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-scheme:sexp-snip get-extent dc x y w h descent space lspace rspace)
⇒ void
  dc : (is-a?/c dc<%>)
  x : real number
  y : real number
  w = #f : boxed non-negative real number or #f
  h = #f : boxed non-negative real number or #f
  descent = #f : boxed non-negative real number or #f
  space = #f : boxed non-negative real number or #f
  lspace = #f : boxed non-negative real number or #f
  rspace = #f : boxed non-negative real number or #f
```

Returns a size corresponding to what this snip draws.

`get-text`

Gets the text representation for this snip.

```
- (send a-scheme:sexp-snip get-text offset num flattened?) ⇒ string
  offset : number
  num : number
  flattened? = #f : boolean
```

Returns the concatenation of the text for all of the hidden snips.

`write`

Writes the snip to the given stream. (Snip reading is handled by the snip class.) Style information about the snip (i.e., the content of `get-style`) will be saved and restored automatically.

¹Many snips cache their size information, so automatically setting the font would be wasteful.

- (send *a-scheme:sexp-snip* write *stream-out*) ⇒ void
 stream-out: `editor-stream-out%`
 Saves the embedded snips

24.3 `scheme:text<%>`

Extends: `text:basic<%>`

Extends: `mode:host-text<%>`

Extends: `color:text<%>`

Texts matching this interface support Scheme mode operations.

`backward-sexp`

Move the caret backwards one sexpression

- (send *a-scheme:text* backward-sexp *start-pos*) ⇒ void
 start-pos: `exact-integer`
 Moves the caret to the beginning of the sexpression that ends at *start-pos*.

`balance-parens`

This function is called when the user types a close parenthesis in the `text%`. If the close parenthesis that the user inserted does not match the corresponding open parenthesis and the `'framework:fixup-parens` preference is `#t` (see `preferences:get`) the correct closing parenthesis is inserted. If the `'framework:paren-match` preference is `#t` (see `preferences:get`) the matching open parenthesis is flashed.

- (send *a-scheme:text* balance-parens *key-event*) ⇒ void
 key-event: (instance `key-event%`)

`box-comment-out-selection`

This method comments out a selection in the text by putting it into a comment box.

- (send *a-scheme:text* box-comment-out-selection *start-pos* *end-pos*) ⇒ void
 start-pos = 'start: (union 'start exact-integer)
 end-pos = 'end: (union 'end exact-integer)

Removes the region from *start-pos* to *end-pos* from the editor and inserts a comment box with that region of text inserted into the box.

If *start-pos* is 'start, the starting point of the selection is used. If *end-pos* is 'end, the ending point of the selection is used.

`comment-out-selection`

- (send *a-scheme:text* `comment-out-selection` *start* *end*) ⇒ void
 start: exact-integer
 end: exact-integer

Comments the lines containing positions *start* through *end* by inserting a semi-colon at the front of each line.

`down-sexp`

- (send *a-scheme:text* `down-sexp` *start*) ⇒ void
 start: exact-integer

Moves forward into the next S-expression after the position *start*.

`find-down-sexp`

- (send *a-scheme:text* `find-down-sexp` *start-pos*) ⇒ (union #f exact-integer)
 start-pos: exact-integer

Returns the position of the beginning of the next sexpression inside the sexpression that contains *start-pos*. If there is no such sexpression, it returns #f.

`find-up-sexp`

- (send *a-scheme:text* `find-up-sexp` *start-pos*) ⇒ (union #f exact-integer)
 start-pos: exact-integer

Returns the position of the beginning of the next sexpression outside the sexpression that contains *start-pos*. If there is no such sexpression, it returns #f.

`flash-backward-sexp`

- (send *a-scheme:text* `flash-backward-sexp` *start-pos*) ⇒ void
 start-pos: exact-integer

Flashes the parenthesis that opens the sexpression at *start-pos*.

`flash-forward-sexp`

- (send *a-scheme:text* `flash-forward-sexp` *start-pos*) ⇒ `void`
start-pos: `exact-integer`

Flashes the parenthesis that closes the sexpression at *start-pos*.

`forward-sexp`

- (send *a-scheme:text* `forward-sexp` *start*) ⇒ `exact-integer`
start: `#t`

Moves forward over the S-expression starting at position *start*.

`get-backward-sexp`

- (send *a-scheme:text* `get-backward-sexp` *start*) ⇒ (union `exact-integer` `#f`)
start: `exact-integer`

Returns the position of the start of the S-expression before or containing *start*, or `#f` if there is no appropriate answer.

`get-forward-sexp`

- (send *a-scheme:text* `get-forward-sexp` *start*) ⇒ (union `#f` `exact-integer`)
start: `exact-integer`

Returns the position of the end of next S-expression after position *start*, or `#f` if there is no appropriate answer.

`get-limit`

- (send *a-scheme:text* `get-limit` *start*) ⇒ `int`
start: `exact-integer`

Returns a limit for backward-matching parenthesis starting at position *start*.

`get-tab-size`

This method returns the current size of the tabs for scheme mode. See also `set-tab-size`.

- (send *a-scheme:text* `get-tab-size`) ⇒ `exact-integer`

`insert-return`

- (send *a-scheme:text* insert-return) ⇒ void

Inserts a newline into the buffer. If `tabify-on-return?` returns #t, this will tabify the new line.

`mark-matching-parenthesis`

If the paren after *pos* is matched, this method highlights it and its matching counterpart in dark green.

- (send *a-scheme:text* mark-matching-parenthesis *pos*) ⇒ void
pos : exact-positive-integer

`remove-parens-forward`

- (send *a-scheme:text* remove-parens-forward *start*) ⇒ void
start : exact-integer

Removes the parentheses from the S-expression starting after the position *start*.

`remove-sexp`

- (send *a-scheme:text* remove-sexp *start*) ⇒ void
start : exact-integer

Forward-deletes the S-expression starting after the position *start*.

`select-backward-sexp`

- (send *a-scheme:text* select-backward-sexp *start*) ⇒ #t
start : exact-integer

Selects the previous S-expression, starting at position *start*.

`select-down-sexp`

- (send *a-scheme:text* select-down-sexp *start*) ⇒ #t
start : exact-integer

Selects the region to the next contained S-expression, starting at position *start*.

`select-forward-sexp`

- (send *a-scheme:text* `select-forward-sexp` *start*) ⇒ #t
start: **exact-integer**

Selects the next S-expression, starting at position *start*.

`select-up-sexp`

- (send *a-scheme:text* `select-up-sexp` *start*) ⇒ #t
start: **exact-integer**

Selects the region to the enclosing S-expression, starting at position *start*.

`set-tab-size`

This method sets the tab size for this text.

- (send *a-scheme:text* `set-tab-size` *new-size*) ⇒ void
new-size: **exact-integer**

`tabify`

- (send *a-scheme:text* `tabify` *start-pos*) ⇒ void
start-pos = (send this `get-start-position`): **exact-integer**

Tabs the line containing by *start-pos*

`tabify-all`

- (send *a-scheme:text* `tabify-all`) ⇒ void

Tabs all lines.

`tabify-on-return?`

The result of this method is used to determine if the return key automatically tabs over to the correct position.

Override it to change it's behavior.

- (send *a-scheme:text* `tabify-on-return?`) ⇒ **boolean**

tabify-selection

- (send *a-scheme:text* tabify-selection *start end*) ⇒ void
 start: exact-integer
 end: exact-integer

Sets the tabbing for the lines containing positions *start* through *end*.

transpose-sexp

- (send *a-scheme:text* transpose-sexp *start*) ⇒ void
 start: exact-integer

Swaps the S-expression beginning before the position *start* with the next S-expression following *start*.

uncomment-selection

- (send *a-scheme:text* uncomment-selection *start end*) ⇒ void
 start: int
 end: int

Uncomments the lines containing positions *start* through *end*.

up-sexp

- (send *a-scheme:text* up-sexp *start*) ⇒ void
 start: exact-integer

Moves backward out of the S-expression containing the position *start*.

24.4 scheme:text-mixin

Domain: `text:basic<%>`

Domain: `mode:host-text<%>`

Domain: `color:text<%>`

Implements: `scheme:text<%>`

Implements: `mode:host-text<%>`

Implements: `color:text<%>`

Implements: `text:basic<%>`

This mixin adds functionality for editing Scheme files.

The result of this mixin uses the same initialization arguments as the mixin's argument.

24.5 `scheme:text-mode<%>`

The result of `scheme:text-mode-mixin` implements this interface.

24.6 `scheme:text-mode-mixin`

Domain: `color:text-mode<%>`

Domain: `mode:surrogate-text<%>`

Implements: `color:text-mode<%>`

Implements: `scheme:text-mode<%>`

Implements: `mode:surrogate-text<%>`

This mixin adds Scheme mode functionality to the mode that it is mixed into. The resulting mode assumes that it is only set to an editor that is the result of `scheme:text-mixin`.

`on-disable-surrogate`

Called when this surrogate is removed from *object*. See also `set-surrogate`.

```
- (send a-scheme:text-mode-mixin on-disable-surrogate object void
  object: object)
```

Removes the scheme keymap (see also `scheme:get-keymap`) and disables any parenthesis highlighting in the host editor.

`on-enable-surrogate`

Called when the surrogate is enabled, with the object it is enabled in. See also `set-surrogate`.

```
- (send a-scheme:text-mode-mixin on-enable-surrogate object void
   object: object)
```

Adds the scheme keymap (see also `scheme:get-keymap`) and enables a parenthesis highlighting in the host editor.

24.7 `scheme:set-mode-mixin`

Domain: `scheme:text<%>`

Domain: `mode:host-text<%>`

Implements: `scheme:text<%>`

Implements: `mode:host-text<%>`

This mixin creates a new instance of

24.8 `text-mode%`

and installs it, by calling its own `set-surrogate` method with the object.

```
24.9 scheme:text% = (mode:host-text-mixin (scheme:set-mode-mixin (scheme:text-mode-mixin
  color:text%)))
```

```
scheme:text% = (mode:host-text-mixin (scheme:set-mode-mixin (scheme:text-mode-mixin
  color:text%)))
```

```
- (new scheme:text% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) => scheme:text%
  object
  line-spacing = 1.0: non-negative real number
  tab-stops = null: list of real numbers
  auto-wrap = #f: boolean
```

Passes all arguments to `super-init`.

```
24.10 scheme:text-mode% = (scheme:text-mode-mixin color:text-mode%)
```

```
scheme:text-mode% = (scheme:text-mode-mixin color:text-mode%)
```

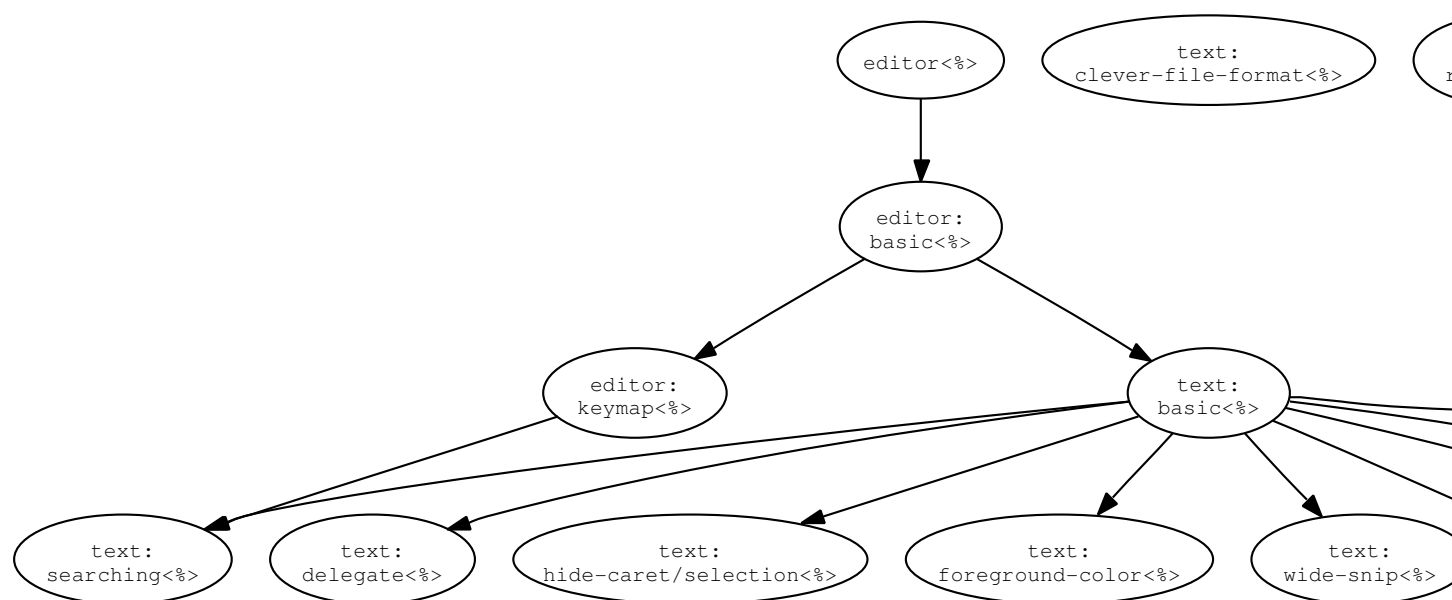
- (new `scheme:text-mode%`) ⇒ `scheme:text-mode%` object

Passes all arguments to `super-init`.

25. Text

This chapter describes the text mixins, interfaces, and classes.

This is the interface hierarchy for the text interface in the framework:



- `text:1-pixel-string-snip%`
- `text:1-pixel-tab-snip%`
- `text:autowrap%`
- `text:backup-autosave%`
- `text:basic%`
- `text:clever-file-format%`
- `text:delegate%`
- `text:file%`

- `text:hide-caret/selection%`
- `text:info%`
- `text:input-box%`
- `text:keymap%`
- `text:nbsp->space%`
- `text:return%`
- `text:searching%`
- `text:standard-style-list%`
- `text:wide-snip%`
- `text:basic<%>`
- `text:clever-file-format<%>`
- `text:delegate<%>`
- `text:file<%>`
- `text:foreground-color<%>`
- `text:hide-caret/selection<%>`
- `text:info<%>`
- `text:input-box<%>`
- `text:nbsp->space<%>`
- `text:ports<%>`
- `text:return<%>`
- `text:searching<%>`
- `text:wide-snip<%>`
- `text:basic-mixin`
- `text:clever-file-format-mixin`
- `text:delegate-mixin`
- `text:file-mixin`
- `text:foreground-color-mixin`
- `text:hide-caret/selection-mixin`
- `text:info-mixin`
- `text:input-box-mixin`
- `text:nbsp->space-mixin`
- `text:ports-mixin`
- `text:return-mixin`
- `text:searching-mixin`
- `text:wide-snip-mixin`

25.1 text:basic<%>

Extends: (class->interface text%)

Extends: editor:basic<%>

Classes matching this interface are expected to implement the basic functionality needed by the framework.

get-fixed-style

Returns the style used by `set-styles-fixed` when setting the styles.

```
- (send a-text:basic get-fixed-style) => (is-a?/c style<%>)
```

get-highlighted-ranges

```
- (send a-text:basic get-highlighted-ranges) => (listof range)
```

Returns a list of (opaque) values representing the active ranges in the editor.

get-styles-fixed

If the result of this function is #t, the styles in this `text:basic<%>` will be fixed. This means that any text inserted to this editor has its style set to this editor's `style-list%`'s "Standard" style.

See also @milink set-styles-fixed

```
- (send a-text:basic get-styles-fixed) => boolean
```

highlight-range

This function highlights a region of text in the buffer.

```
- (send a-text:basic highlight-range start end color bitmap caret-space priority)
=> (-: void)
  start: exact-integer
  end: exact-integer
  color: (instance color%)
  bitmap = #f: (union #f (instance bitmap%))
```

```

caret-space = #f : boolean
priority = 'low : (union 'high 'low)

```

The range between *start* and *end* will be highlighted with the color in *color*, and *bitmap* will be painted over the range of text in black and white. If *bitmap* is #f, the range will be inverted, using the platform specific xor. This method is not recommended, because the selection is also displayed using xor.

If *caret-space?* is not #f, the left edge of the range will be one pixel short, to leave space for the caret. The caret does not interfere with the right hand side of the range. Note that under X windows the caret is drawn with XOR, which means almost anything can happen. So if the caret is in the middle of the range it may be hard to see, or if it is on the left of the range and *caret-space?* is #f it may also be hard to see.

The *priority* argument indicates the relative priority for drawing overlapping regions. If two regions overlap and have different priorities, the region with 'high priority will be drawn second and only it will be visible in the overlapping region.

This method returns a thunk, which, when invoked, will turn off the highlighting from this range.

initial-autowrap-bitmap

The result of this method is used as the initial autowrap bitmap. Override this method to change the initial `bitmap%`. See also `set-autowrap-bitmap`

```

- (send a-text:basic initial-autowrap-bitmap) => (union #f (instance bitmap%))
  Defaults returns the result of icon:get-autowrap-bitmap

```

move/copy-to-edit

This moves or copies text and snips to another edit.

```

- (send a-text:basic move/copy-to-edit dest-text start end dest-pos) => void
  dest-text : (instance text%)
  start : exact-integer
  end : exact-integer
  dest-pos : exact-integer

```

Moves or copies from the edit starting at *start* and ending at *end*. It puts the copied text and snips in *dest-text* starting at location *dest-pos*.

If a snip refused to be moved, it will be copied, otherwise it will be moved. A snip may refuse to be moved by returning #f from `release-from-owner`.

set-styles-fixed

Sets the styles fixed parameter of this `text%`. See also `get-styles-fixed` and `get-fixed-style`.

```

- (send a-text:basic set-styles-fixed fixed?) => void
  fixed? : boolean

```

25.2 text:basic-mixin

Domain: `editor:basic<%>`

Domain: (class->interface `text%`)

Implements: `editor:basic<%>`

Implements: `text:basic<%>`

This mixin implements the basic functionality needed for `text%` objects in the framework.

The class that this mixin produces uses the same initialization arguments as it's input.

```
- init args: [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]
  line-spacing=1.0: non-negative real number
  tab-stops=null: list of real numbers
  auto-wrap=#f: boolean
```

The `line-spacing` argument sets the additional amount of space (in DC units) inserted between each line in the editor when the editor is displayed. This spacing is included in the reported height of each line.

See `set-tabs` for information about `tabstops`.

If `auto-wrap` is true, then auto-wrapping is enabled via `auto-wrap`.

A new `keymap%` object is created for the new editor. See also `get-keymap` and `set-keymap`.

A new `style-list%` object is created for the new editor. See also `get-style-list` and `set-style-list`.

`after-insert` (*augments, and augmentable only*)

Called after `items` are inserted into the editor (and after the `display` is refreshed; use `on-insert` and `begin-edit-sequence` to avoid extra refreshes when `after-insert` modifies the editor).

See also `can-insert?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```
- (send a-text:basic-mixin after-insert start len void
  start: exact non-negative integer
  len: exact non-negative integer
  See set-styles-fixed.
```

`on-insert` (*augments, and augmentable only*)

Called before `items` are inserted into the editor, after `can-insert?` is called to verify that the insertion is ok. The `after-insert` method is guaranteed to be called after the insert has completed.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-insert` to modify the editor, if necessary.

See also `on-edit-sequence`.

```
- (send a-text:basic-mixin on-insert start end void
    start : exact-int
    end : exact-int
```

See `set-styles-fixed`.

`on-paint`

Provides a way to add arbitrary graphics to an editor’s `display`. This method is called just before and just after every painting of the editor.

The `on-paint` method, together with the snips’ `draw` methods, must be able to draw the entire state of an editor. Never paint directly into an editor’s `display` canvas except from within `on-paint` or `draw`. Instead, put all extra drawing code within `on-paint` and call `invalidate-bitmap-cache` when part of the `display` needs to be repainted.

If an `on-paint` method uses cached `location` information, then the cached information should be recomputed in response to a call of `invalidate-bitmap-cache`.

The `on-paint` method must not make any assumptions about the state of the drawing context (e.g., the current pen), except that the clipping region is already set to something appropriate. Before `on-paint` returns, it must restore any drawing context settings that it changes.

The editor is internally locked for writing and reflowing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-text:basic-mixin on-paint before? dc left top right bottom dx dy draw-caret
    void
    before? : boolean
    dc : dc<%> object
    left : real number
    top : real number
    right : real number
    bottom : real number
    dx : real number
    dy : real number
    draw-caret : symbol in '(no-caret show-inactive-caret show-caret)
```

Draws the rectangles installed by `highlight-range`.

25.3 `text:foreground-color<%>`

Extends: `text:basic<%>`

Extends: `editor:standard-style-list<%>`

25.4 text:foreground-color-mixin

Domain: `text:basic<%>`

Domain: `editor:standard-style-list<%>`

Implements: `text:basic<%>`

Implements: `text:foreground-color<%>`

Implements: `editor:standard-style-list<%>`

This mixin changes the default text style to have the foreground color controlled by `editor:set-default-font-color`.

`get-fixed-style`

Returns the style used by `set-styles-fixed` when setting the styles.

```
- (send a-text:foreground-color-mixin get-fixed-style (is-a?/c style<%>))
```

Returns the style named by `editor:get-default-color-style-name`.

25.5 text:hide-caret/selection<%>

Extends: `text:basic<%>`

This class hides the caret, except when the selection is active.

Instances of this class are useful for editors that used for displaying purposes, but still allow users to copy their text.

25.6 text:hide-caret/selection-mixin

Domain: `text:basic<%>`

Implements: `text:basic<%>`

Implements: `text:hide-caret/selection<%>`

`after-set-position` (*augments, and augmentable only*)

Called after the start and end `position` have been moved (but not when the `position` is moved due to inserts or deletes).

See also `on-edit-sequence`.

```
- (send a-text:hide-caret/selection-mixin after-set-position void
  Calls hide-caret to hide the caret when there is only a caret and no selection.
```

25.7 text:nbsp->space<%>

Extends: (class->interface `text%`)

Classes that implement this interface silently change non-breaking spaces, ie the character (integer->char 160), to regular spaces when inserted into the editor.

25.8 text:nbsp->space-mixin

Domain: (class->interface `text%`)

Implements: `text:nbsp->space<%>`

```
- init args: [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]
  line-spacing = 1.0: non-negative real number
  tab-stops = null: list of real numbers
  auto-wrap = #f: boolean
```

The `line-spacing` argument sets the additional amount of space (in DC units) inserted between each line in the editor when the editor is displayed. This spacing is included in the reported height of each line.

See `set-tabs` for information about `tabstops`.

If `auto-wrap` is true, then auto-wrapping is enabled via `auto-wrap`.

A new `keymap%` object is created for the new editor. See also `get-keymap` and `set-keymap`.

A new `style-list%` object is created for the new editor. See also `get-style-list` and `set-style-list`.

`after-insert` (*augments, and augmentable only*)

Called after `items` are inserted into the editor (and after the `display` is refreshed; use `on-insert` and `begin-edit-sequence` to avoid extra refreshes when `after-insert` modifies the editor).

See also `can-insert?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```
- (send a-text:nbsp->space-mixin after-insert start len void
  start: exact non-negative integer
  len: exact non-negative integer
```

Replaces all non-breaking space characters

```
(integer->char 160)
```

by `#\space` characters.

Ends the edit sequence (by calling `end-edit-sequence`) started in `on-insert`.

`on-insert` (*augments, and augmentable only*)

Called before `items` are inserted into the editor, after `can-insert?` is called to verify that the insertion is ok. The `after-insert` method is guaranteed to be called after the insert has completed.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-insert` to modify the editor, if necessary.

See also `on-edit-sequence`.

```
- (send a-text:nbsp->space-mixin on-insert start end void
  start: exact-int
  end: exact-int
```

Starts an edit-sequence by calling `begin-edit-sequence`.

25.9 text:searching<%>

Extends: `text:basic<%>`

Extends: `editor:keymap<%>`

Any object matching this interface can be searched.

25.10 text:searching-mixin

Domain: `text:basic<%>`

Domain: `editor:keymap<%>`

Implements: `text:basic<%>`

Implements: `text:searching<%>`

Implements: `editor:keymap<%>`

This `text%` can be searched.

The result of this mixin uses the same initialization arguments as the mixin's argument.

```
get-keymaps
```

The keymaps returned from this method are chained to this `editor<%>`'s keymap.

```
- (send a-text:searching-mixin get-keymaps (list-of (instance keymap%)))
```

This returns a list containing the super-class's keymaps, plus the result of `keymap:get-search`

25.11 text:return<%>

Extends: (class->interface `text%`)

Objects supporting this interface were created by `text:return-mixin`.

25.12 text:return-mixin

Domain: (class->interface `text%`)

Implements: `text:return<%>`

Use this buffer to perform some special action when return is typed.

```
- init args: return
  return: (-¿ boolean)
```

on-local-char

Called by `on-char` when the event is *not* handled by a caret-owning snip.

Consider overriding `on-default-char` instead of this method.

```
- (send a-text:return-mixin on-local-char event void
  event: key-event% object
```

If *key* is either return or newline, only invoke the *return* thunk (initialization argument) and do nothing else.

25.13 text:wide-snip<%>

Extends: `text:basic<%>`

add-tall-snip

Registers a snip in this editor. It is resized when the viewing area of the editor changes.

This method should only be called by `add-tall-snip`.

```
- (send a-text:wide-snip add-tall-snip snip) ⇒ void
  snip: (is-a?/c snip%)
```

add-wide-snip

Registers a snip in this editor to be resized when its viewing area changes. Ensures the snip is as wide as the viewing area.

This method should only be called by `add-tall-snip`.

```
- (send a-text:wide-snip add-wide-snip snip) ⇒ void
  snip: (instance snip%)
```

25.14 text:wide-snip-mixin

Domain: `text:basic<%>`

Implements: `text:basic<%>`

Implements: `text:wide-snip<%>`

25.15 `text:delegate<%>`

Extends: `text:basic<%>`

Implementations of this interface copy all of the changes to this editor to the result of `get-delegate` except instead of regular string and tab snips, instead instances of `text:1-pixel-string-snip%` and `text:1-pixel-tab-snip%` are created.

The contents of the two editor are kept in sync, as modifications to this object happen.

`get-delegate`

The result of this method is the `text%` object that the contents of this editor are being delegated to, or `#f`, if there is none.

```
- (send a-text:delegate get-delegate) => (union #f (instanceof text%))
```

`set-delegate`

This method sets the current delegate.

```
- (send a-text:delegate set-delegate delegate) => void
  delegate: (union #f (instanceof text%))
```

When it is set, all of the snips are copied from this object to `delegate`. Additionally, if this object implements `scheme:text<%>` the tab settings of `delegate` are updated to match this objects.

25.16 `text:1-pixel-string-snip%`

Superclass: `string-snip%`

This class re-uses the implementation of `string-snip%` to implement a string snip that just draws a single pixel for each character in the string.

See also `text:1-pixel-tab-snip%` for a similar extension to the `tab-snip%` class.

This snip is used in conjunction with the `frame:delegate<%>` and `text:delegate<%>` interfaces.

- (make-object text:1-pixel-string-snip% allocsize) ⇒ text:1-pixel-string-snip% object
 allocsize = 0: exact non-negative integer
 Creates an empty string snip. The *allocsize* argument is a hint about how much storage space for text should be initially allocated by the snip.
- (make-object text:1-pixel-string-snip% s) ⇒ text:1-pixel-string-snip% object
 s: string
 Creates a string snip with the given initial string.

copy

Creates and returns a copy of this snip. The `copy` method is responsible for copying this snip's style (as returned by `get-style`) to the new snip.

- (send a-text:1-pixel-string-snip copy) ⇒ (instanceof snip%)
 Creates and returns an instance of `text:1-pixel-string-snip%`.

draw

Called (by an editor) to draw the snip.

Before this method is called, the correct font, text color, and pen color will have been set in the drawing context for this snip already. (This is *not* true for `get-extent` or `partial-offset`.) The `draw` method must not make any other assumptions about the state of the drawing context, except that the clipping region is already set to something appropriate. Before `draw` returns, it must restore any drawing context settings that it changes.

See also `on-paint in editor<%>`.

The snip's editor is usually internally locked for writing and reflowing when this method is called (see also "Locks" (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

- (send a-text:1-pixel-string-snip draw dc x y left top right bottom dx dy draw-caret)
 ⇒ void
 dc: (instanceof dc<%>)
 x: real number
 y: real number
 left: real number
 top: real number
 right: real number
 bottom: real number
 dx: real number
 dy: real number
 draw-caret: (union 'no-caret 'show-inactive-caret 'show-caret)

Draws black pixels for non-whitespace characters and draws nothing for whitespace characters.

`get-extent`

Calculates the snip’s width, height, descent (amount of height which is drawn below the baseline), space (amount of height which is “filler” space at the top), and horizontal spaces (amount of width which is “filler” space at the left and right).

This method is called by the snip’s administrator; it should not be called directly by others. To get the extent of a snip, use `get-snip-location in editor<%>`.

A drawing context is provided for the purpose of finding font sizes, but no drawing should occur. The `get-extent` and `partial-offset` methods must not make any assumptions about the state of the drawing context, except that it is scaled properly. In particular, the font for the snip’s style is not automatically set in the drawing context before the method is called.¹ If `get-extent` or `partial-offset` changes the drawing context’s setting, it must restore them before returning. However, the methods should not need to change the drawing context; only font settings can affect measurement results from a device context, and `get-text-extent in dc<%>` accepts a `font%` argument for sizing that overrides that device context’s current font.

The snip’s left and top `locations` are provided in editor coordinates. In a text editor, the y-coordinate is the *line*’s top `location`; the snip’s actual top `location` is potentially undetermined until its height is known.

If a snip caches the result size for future replies, it should invalidate its cached size when `size-cache-invalid` is called (especially if the snip’s size depends on any device context properties).

If a snip’s size changes after receiving a call to `get-extent` and before receiving a call to `size-cache-invalid`, then the snip must notify its administrator of the size change, so that the administrator can recompute its derived size information. Notify the administrator of a size change by call its `resized` method.

The snip’s editor is usually internally locked for writing and reflowing when this method is called (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-text:1-pixel-string-snip get-extent dc x y w h descent space lspace
  rspace) ⇒ void
  dc : (instanceof dc<%>)
  x : real number
  y : real number
  w = #f : (box (union non-negative-real-number #f))
  h = #f : (box (union non-negative-real-number #f))
  descent = #f : (box (union non-negative-real-number #f))
  space = #f : (box (union non-negative-real-number #f))
  lspace = #f : (box (union non-negative-real-number #f))
  rspace = #f : (box (union non-negative-real-number #f))
```

Sets the descent, space, lspace, and rspace to zero. Sets the height to 1. Sets the width to the number of characters in the string.

`insert`

¹Many snips cache their size information, so automatically setting the font would be wasteful.

Inserts text into the snip. The system can insert text into a text snip without calling this method.

```
- (send a-text:1-pixel-string-snip insert s len pos) ⇒ void
  s : string
  len : exact non-negative integer
  pos = 0 : exact non-negative integer
```

split

Splits the snip into two snips. This is called when a snip has more than one **item** and something is inserted between two **items**.

The arguments are a relative **position** integer and two boxes. The **position** integer specifies how many **items** should be given to the new first snip; the rest go to the new second snip. The two boxes must be filled with two new snips. (The old snip is no longer used, so it can be recycled as a new snip.)

If the returned snips do not have the expected **counts**, their **counts** are forcibly modified. If either returned snip is already owned by another administrator, a surrogate snip is created.

The snip's editor is usually internally locked for reading when this method is called (see also "Locks" (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-text:1-pixel-string-snip split position first second) ⇒ void
  position : exact non-negative integer
  first : (box (instanceof snip%))
  second : (box (instanceof snip%))
```

Fills the boxes with instance of **text:1-pixel-string-snip%**.

25.17 text:1-pixel-tab-snip%

Superclass: **tab-snip%**

This class re-uses the implementation of **tab-snip%** to implement a string snip that is always one pixel high.

See also **text:1-pixel-string-snip%** for a similar extension to the **string-snip%** class.

This snip is used in conjunction with the **frame:delegate<%>** and **text:delegate<%>** interfaces.

```
- (make-object text:1-pixel-tab-snip% allocsize s) ⇒ text:1-pixel-tab-snip%
  object
  allocsize = 0 : exact non-negative integer
  s : string
```

Creates a snip for a single tab, though the tab is initially empty.

Normally, a single tab is inserted into a **tab-snip%** object using the **insert** method.

The tab's content is not drawn, through it is used when determining the size of a single character in editors where tabbing is determined by the character width (see **set-tabs**); if the content is a single tab character (the normal case), then the average character width of snip's font is used as the tab's width.

`copy`

Creates and returns a copy of this snip. The `copy` method is responsible for copying this snip's style (as returned by `get-style`) to the new snip.

```
- (send a-text:1-pixel-tab-snip copy) ⇒ (instanceof snip%)
```

Creates and returns an instance of `text:1-pixel-tab-snip%`.

`draw`

Called (by an editor) to draw the snip.

Before this method is called, the correct font, text color, and pen color will have been set in the drawing context for this snip already. (This is *not* true for `get-extent` or `partial-offset`.) The `draw` method must not make any other assumptions about the state of the drawing context, except that the clipping region is already set to something appropriate. Before `draw` returns, it must restore any drawing context settings that it changes.

See also `on-paint in editor<%>`.

The snip's editor is usually internally locked for writing and reflowing when this method is called (see also "Locks" (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-text:1-pixel-tab-snip draw dc x y left top right bottom dx dy draw-caret)
⇒ void
  dc : (instanceof dc<%>)
  x : real number
  y : real number
  left : real number
  top : real number
  right : real number
  bottom : real number
  dx : real number
  dy : real number
  draw-caret : (union 'no-caret 'show-inactive-caret 'show-caret)
```

Draws nothing.

`get-extent`

Calculates the snip's width, height, descent (amount of height which is drawn below the baseline), space (amount of height which is "filler" space at the top), and horizontal spaces (amount of width which is "filler" space at the left and right).

This method is called by the snip's administrator; it should not be called directly by others. To get the extent of a snip, use `get-snip-location in editor<%>`.

A drawing context is provided for the purpose of finding font sizes, but no drawing should occur. The `get-extent` and `partial-offset` methods must not make any assumptions about the state of the drawing context, except that

it is scaled properly. In particular, the font for the snip’s style is not automatically set in the drawing context before the method is called.² If `get-extent` or `partial-offset` changes the drawing context’s setting, it must restore them before returning. However, the methods should not need to change the drawing context; only font settings can affect measurement results from a device context, and `get-text-extent in dc<%>` accepts a `font%` argument for sizing that overrides that device context’s current font.

The snip’s left and top `location`s are provided in editor coordinates. In a text editor, the y-coordinate is the *line*’s top `location`; the snip’s actual top `location` is potentially undetermined until its height is known.

If a snip caches the result size for future replies, it should invalidate its cached size when `size-cache-invalid` is called (especially if the snip’s size depends on any device context properties).

If a snip’s size changes after receiving a call to `get-extent` and before receiving a call to `size-cache-invalid`, then the snip must notify its administrator of the size change, so that the administrator can recompute its derived size information. Notify the administrator of a size change by call its `resized` method.

The snip’s editor is usually internally locked for writing and reflowing when this method is called (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-text:1-pixel-tab-snip get-extent dc x y w h descent space lspace rspace)
⇒ void
  dc : (instanceof dc<%>)
  x : real number
  y : real number
  w = #f : (box (union non-negative-real-number #f))
  h = #f : (box (union non-negative-real-number #f))
  descent = #f : (box (union non-negative-real-number #f))
  space = #f : (box (union non-negative-real-number #f))
  lspace = #f : (box (union non-negative-real-number #f))
  rspace = #f : (box (union non-negative-real-number #f))

Sets the descent, space, lspace, and rspace to zero. Sets the height to 1. Sets the width to the width of tabs as
returned in the tab-width parameter of the get-tabs method.
```

`split`

Splits the snip into two snips. This is called when a snip has more than one `item` and something is inserted between two `items`.

The arguments are a relative `position` integer and two boxes. The `position` integer specifies how many `items` should be given to the new first snip; the rest go to the new second snip. The two boxes must be filled with two new snips. (The old snip is no longer used, so it can be recycled as a new snip.)

If the returned snips do not have the expected `counts`, their `counts` are forcibly modified. If either returned snip is already owned by another administrator, a surrogate snip is created.

The snip’s editor is usually internally locked for reading when this method is called (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-text:1-pixel-tab-snip split position first second) ⇒ void
  position: exact non-negative integer
```

²Many snips cache their size information, so automatically setting the font would be wasteful.

```

first: (box (instanceof snip%))
second: (box (instanceof snip%))

```

Fills the boxes with instance of `text:1-pixel-tab-snip%`.

25.18 text:delegate-mixin

Domain: `text:basic<%>`

Implements: `text:basic<%>`

Implements: `text:delegate<%>`

This mixin provides an implementation of the `text:delegate<%>` interface.

`after-change-style` (*augments, and augmentable only*)

Called after the style is changed for a given range (and after the `display` is refreshed; use `on-change-style` and `begin-edit-sequence` to avoid extra refreshes when `after-change-style` modifies the editor).

See also `can-change-style?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```

- (send a-text:delegate-mixin after-change-style start len void
  start: number
  len: number

```

forwards the changed style to the delegate.

`after-delete` (*augments, and augmentable only*)

Called after a given range is deleted from the editor (and after the `display` is refreshed; use `on-delete` and `begin-edit-sequence` to avoid extra refreshes when `after-delete` modifies the editor).

See also `can-delete?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```

- (send a-text:delegate-mixin after-delete start len void
  start: number
  len: number

```

forwards the change to the delegate.

after-edit-sequence (*augments, and augmentable only*)

Called after a top-level edit sequence completes (involving unnested `begin-edit-sequence` and `end-edit-sequence`).

See also `on-edit-sequence`.

- (send `a-text:delegate-mixin` after-edit-sequence void
ends an edit sequence in the delegate.

after-insert (*augments, and augmentable only*)

Called after `items` are inserted into the editor (and after the `display` is refreshed; use `on-insert` and `begin-edit-sequence` to avoid extra refreshes when `after-insert` modifies the editor).

See also `can-insert?` and `on-edit-sequence`.

No internals locks are set when this method is called.

- (send `a-text:delegate-mixin` after-insert `start len` void
 `start`: number
 `len`: number
forwards the change to the delegate

after-load-file (*augments, and augmentable only*)

Called just after the editor is loaded from a file.

The argument to the method originally specified whether the save was successful, but failures now trigger exceptions such that the method is not even called. Consequently, the argument is always `#t`.

See also `can-load-file?` and `on-load-file`.

- (send `a-text:delegate-mixin` after-load-file `success?` void
 `success?`: boolean
updates the delegate with the new contents of the text.

highlight-range

This function highlights a region of text in the buffer.

- (send `a-text:delegate-mixin` highlight-range `start end color bitmap caret-space`
 `priority` (-? void)
 `start`: exact-integer

```

end: exact-integer
color: (instance color%)
bitmap: (union #f (instance bitmap%))
caret-space=#f: boolean
priority='low: (union 'high 'low)

```

In addition to calling the super method, `highlight-range`, this method forwards the highlighting to the delegatee.

`on-edit-sequence` (*augments, and augmentable only*)

Called just after a top-level (i.e., unnested) edit sequence starts.

During an edit sequence, all callback methods are invoked normally, but it may be appropriate for these callbacks to delay computation during an edit sequence. The callbacks must manage this delay manually. Thus, when overriding other callback methods, such as `on-insert in text%`, `on-insert in pasteboard%`, `after-insert in text%`, or `after-insert in pasteboard%`, consider overriding `on-edit-sequence` and `after-edit-sequence` as well.

“Top-level edit sequence” refers to an outermost pair of `begin-edit-sequence` and `end-edit-sequence` calls. The embedding of an editor within another editor does not affect the timing of calls to `on-edit-sequence`, even if the embedding editor is in an edit sequence.

Pairings of `on-edit-sequence` and `after-edit-sequence` can be nested if an `after-edit-sequence` starts a new edit sequence, since `after-edit-sequence` is called after an edit sequence ends. However, `on-edit-sequence` can never start a new top-level edit sequence (except through an unpaired `end-edit-sequence`), because it is called after a top-level edit sequence starts.

```

- (send a-text:delegate-mixin on-edit-sequence void
  starts an edit sequence in the delegate.

```

`on-load-file` (*augments, and augmentable only*)

Called just before the editor is loaded from a file, after calling `can-load-file?` to verify that the load is allowed. See also `after-load-file`.

```

- (send a-text:delegate-mixin on-load-file filename format void
  filename: string
  format: symbol
  remembers the filename, for use in after-load-file.

```

`on-paint`

Provides a way to add arbitrary graphics to an editor’s `display`. This method is called just before and just after every painting of the editor.

The `on-paint` method, together with the snips' `draw` methods, must be able to draw the entire state of an editor. Never paint directly into an editor's `display` canvas except from within `on-paint` or `draw`. Instead, put all extra drawing code within `on-paint` and call `invalidate-bitmap-cache` when part of the `display` needs to be repainted.

If an `on-paint` method uses cached `location` information, then the cached information should be recomputed in response to a call of `invalidate-bitmap-cache`.

The `on-paint` method must not make any assumptions about the state of the drawing context (e.g., the current pen), except that the clipping region is already set to something appropriate. Before `on-paint` returns, it must restore any drawing context settings that it changes.

The editor is internally locked for writing and reflowing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)).

```
- (send a-text:delegate-mixin on-paint before? dc left top right bottom dx dy
draw-caret void
before?: boolean
dc: dc<%> object
left: real number
top: real number
right: real number
bottom: real number
dx: real number
dy: real number
draw-caret: symbol in '(no-caret show-inactive-caret show-caret)
```

Draws a blue region in the delegatee editor that shows where the visible region of the delegate editor is.

resized

Called (indirectly) by snips within the editor: it forces a recalculation of the display information in which the specified snip has changed its size.

```
- (send a-text:delegate-mixin resized snip redraw-now? void
snip: (is-a?/c snip%)
redraw-now?: boolean
```

Sends a message to the delegate to update the size of the copied snip, if there is one.

25.19 text:info<%>

Extends: `text:basic<%>`

Objects supporting this interface are expected to send information about themselves to the frame that is displaying them.

25.20 text:info-mixin

Domain: `text:basic<%>`

Domain: `editor:keymap<%>`

Implements: `text:basic<%>`

Implements: `editor:keymap<%>`

Implements: `text:info<%>`

This mixin adds support for supplying information to objects created with `frame:info-mixin`. When this `editor:basic<%>` is displayed in a frame, that frame must have been created with `frame:info-mixin`.

`after-delete` (*augments, and augmentable only*)

Called after a given range is deleted from the editor (and after the `display` is refreshed; use `on-delete` and `begin-edit-sequence` to avoid extra refreshes when `after-delete` modifies the editor).

See also `can-delete?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```
- (send a-text:info-mixin after-delete start len void
  start: exact non-negative integer
  len: exact non-negative integer
```

Calls the `editor-position-changed` method of the frame that is viewing this object. It uses `get-canvas` to get the canvas for this frame, and uses that canvas's `top-level-window<%>` as the frame.

`after-insert` (*augments, and augmentable only*)

Called after `items` are inserted into the editor (and after the `display` is refreshed; use `on-insert` and `begin-edit-sequence` to avoid extra refreshes when `after-insert` modifies the editor).

See also `can-insert?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```
- (send a-text:info-mixin after-insert start len void
  start: exact non-negative integer
  len: exact non-negative integer
```

Calls the `editor-position-changed` method of the frame that is viewing this object. It uses `get-canvas` to get the canvas for this frame, and uses that canvas's `top-level-window<%>` as the frame.

after-set-position (*augments, and augmentable only*)

Called after the start and end **position** have been moved (but not when the **position** is moved due to inserts or deletes).

See also [on-edit-sequence](#).

```
- (send a-text:info-mixin after-set-position void
```

Calls the [editor-position-changed](#) method of the frame that is viewing this object. It uses [get-canvas](#) to get the canvas for this frame, and uses that canvas's [top-level-window<%>](#) as the frame.

set-anchor

Turns anchoring on or off. This method can be overridden to affect or detect changes in the anchor state. See also [get-anchor](#).

```
- (send a-text:info-mixin set-anchor on? void
  on?: boolean
```

Calls the [anchor-status-changed](#) method of the frame that is viewing this object. It uses [get-canvas](#) to get the canvas for this frame, and uses that canvas's [top-level-window<%>](#) as the frame.

set-overwrite-mode

Enables or disables overwrite mode. See [get-overwrite-mode](#). This method can be overridden to affect or detect changes in the overwrite mode.

```
- (send a-text:info-mixin set-overwrite-mode on? void
  on?: boolean
```

Calls the [overwrite-status-changed](#) method of the frame that is viewing this object. It uses [get-canvas](#) to get the canvas for this frame, and uses that canvas's [top-level-window<%>](#) as the frame.

25.21 text:clever-file-format<%>

Extends: (class->interface [text%](#))

Objects supporting this interface are expected to support a clever file format when saving.

25.22 text:clever-file-format-mixin

Domain: (class->interface text%)

Implements: text:clever-file-format<%>

The result of this mixin uses the same initialization arguments as the mixin's argument.

When files are saved from this text%, a check is made to see if there are any non-string-snip% objects in the text%. If so, it is saved using the file format 'std. (see set-file-format for more information. If not, the file format passed to save-file is used.

```
- init args: [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]
  line-spacing = 1.0 : non-negative real number
  tab-stops = null : list of real numbers
  auto-wrap = #f : boolean
```

The *line-spacing* argument sets the additional amount of space (in DC units) inserted between each line in the editor when the editor is displayed. This spacing is included in the reported height of each line.

See `set-tabs` for information about *tabstops*.

If *auto-wrap* is true, then auto-wrapping is enabled via `auto-wrap`.

A new `keymap%` object is created for the new editor. See also `get-keymap` and `set-keymap`.

A new `style-list%` object is created for the new editor. See also `get-style-list` and `set-style-list`.

on-save-file (*augments, and augmentable only*)

Called just before the editor is saved to a file, after calling `can-save-file?` to verify that the save is allowed. See also `after-save-file`.

```
- (send a-text:clever-file-format-mixin on-save-file filename format void
  filename: path
  format: symbol in '(guess standard text text-force-cr same copy)
```

If the method `get-file-format` returns 'standard and the text has only string-snip%s, the file format is set to 'text.

If the method `get-file-format` returns 'text and the text has some non string-snip%s, the file format is set to 'standard.

Depending on the user's preferences, the user may also be queried.

Also, the changes to the file format only happen if the argument *file-format* is 'copy or 'same.

25.23 text:file<%>

Extends: text:basic<%>

Extends: editor:file<%>

Mixins that implement this interface lock themselves when the file they are editing is read only.

get-read-write?

Indicates whether or not this editor is in read-write mode.

```
- (send a-text:file get-read-write?) => boolean
```

25.24 text:file-mixin

Domain: `text:basic<%>`

Domain: `editor:file<%>`

Implements: `text:basic<%>`

Implements: `text:file<%>`

Implements: `editor:file<%>`

after-load-file (*augments, and augmentable only*)

Called just after the editor is loaded from a file.

The argument to the method originally specified whether the save was successful, but failures now trigger exceptions such that the method is not even called. Consequently, the argument is always #t.

See also `can-load-file?` and `on-load-file`.

```
- (send a-text:file-mixin after-load-file void
  Checks if the newly loaded file is write-only in the filesystem. If so, locks the editor with the lock method.
  Otherwise unlocks the buffer
  For each canvas returned from get-canvass it checks to see if the canvas%'s get-top-level-window
  matches the frame:editor<%> interface. If so, it calls set-label with the last part of the filename (ie,
  the name of the file, not the directory the file is in).
```

after-save-file (*augments, and augmentable only*)

Called just after the editor is saved to a file.

The argument to the method originally specified whether the save was successful, but failures now trigger exceptions such that the method is not even called. Consequently, the argument is always `#t`.

See also `can-save-file?` and `on-save-file`.

```
- (send a-text:file-mixin after-save-file void
```

Checks if the newly saved file is write-only in the filesystem. If so, locks the editor with the `lock` method. Otherwise unlocks the buffer

For each canvas returned from `get-canvases` it checks to see if the `canvas`'s `get-top-level-window` matches the `frame:editor<%>` interface. If so, it calls `set-label` with the last part of the filename (ie, the name of the file, not the directory the file is in).

`can-delete?` (*augments, and augmentable only*)

Called before a range is deleted from the editor. If the return value is `#f`, then the delete will be aborted.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-delete` to modify the editor, if necessary.

See also `on-delete`, `after-delete`, and `on-edit-sequence`.

```
- (send a-text:file-mixin can-delete? start len boolean
  start: number
  len: number
```

Returns false if the result of `get-read-write?` is true, otherwise returns the result of calling `inner`.

`can-insert?` (*augments, and augmentable only*)

Called before `items` are inserted into the editor. If the return value is `#f`, then the insert will be aborted.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-insert` to modify the editor, if necessary.

See also `on-insert`, `after-insert`, and `on-edit-sequence`.

```
- (send a-text:file-mixin can-insert? start len boolean
  start: number
  len: number
```

Returns false if the result of `get-read-write?` is true, otherwise returns the result of calling `inner`.

25.25 `text:ports<%>`

Classes implementing this interface (via the associated mixin) support input and output ports that read from the editor.

There are two input ports: the normal input port just reads from the editor's contents directly and the box input port inserts an editor snip into this text and uses input typed into the box as input into the port.

They create three threads to mediate access to the input and output ports (one for each input port and one for all of the output ports).

after-io-insertion

This method is called after an insertion due to IO occurs.

```
- (send a-text:ports after-io-insertion) ⇒ void
```

clear-box-input-port

Flushes all of the data in the box input port that hasn't yet been read. Reading will now block.

```
- (send a-text:ports clear-box-input-port) ⇒ void
```

clear-input-port

Flushes all of the data in the input port that hasn't yet been read. Reading will now block.

```
- (send a-text:ports clear-input-port) ⇒ void
```

clear-output-ports

Flushes all of the data in all of the output ports that hasn't appeared in the editor yet.

```
- (send a-text:ports clear-output-ports) ⇒ void
```

delete/io

Deletes the text between *start* and *end* without changing the behavior of the ports (otherwise, deleting the text would break internal invariants of the port).

Both *start* and *end* must be less than `get-insertion-point` (or else it is safe to delete them so you don't need this method).

```
- (send a-text:ports delete/io start end) ⇒ void
  start: exact-integer
  end: exact-integer
```

`get-allow-edits`

Indicates if editing is allowed in the the buffer at this point.

```
- (send a-text:ports get-allow-edits) ⇒ boolean
```

`get-box-input-editor-snip%`

The result of this method is used as the class of editor snips that is inserted by the box port in this editor.

```
- (send a-text:ports get-box-input-editor-snip%) ⇒ (subclass editor-snip%)
```

`get-box-input-text%`

The result of this method is instantiated and placed inside the result of `get-box-input-editor-snip%`.

```
- (send a-text:ports get-box-input-text%) ⇒ (implements text:input-box)
```

`get-err-port`

Returns an output port that writes into this editor. The only difference between this port and the ports returned by `get-err-port` and `get-out-port` is the font style and color.

```
- (send a-text:ports get-err-port) ⇒ output-port
```

`get-err-style-delta`

The result of this method is the style that is used to color text submitted to the result of `get-err-port`.

This method is called during the initialization of the class.

```
- (send a-text:ports get-err-style-delta) ⇒ (is-a?/c style-delta%)
```

Returns a style delta that uses the default font (not a fixed-width font) and colors the text red and makes it italic.

`get-in-box-port`

Returns the box input port that data in this editor is sent to.

```
- (send a-text:ports get-in-box-port) ⇒ input-port
```

get-in-port

Returns the input port that data in this editor is sent to.

```
- (send a-text:ports get-in-port) ⇒ input-port
```

get-insertion-point

Returns the position where characters put into the output port will appear.

```
- (send a-text:ports get-insertion-point) ⇒ exact-integer
```

get-out-port

Returns an output port that writes into this editor. The only difference between this port and the ports returned by `get-err-port` and `get-value-port` is the font style and color.

```
- (send a-text:ports get-out-port) ⇒ output-port
```

get-out-style-delta

The result of this method is the style that is used to color text submitted to the result of `get-out-port`.

This method is called during the initialization of the class.

```
- (send a-text:ports get-out-style-delta) ⇒ (is-a?/c style-delta%)
```

Returns a style delta that uses the default font (not a fixed-width font) and colors the text purple.

get-unread-start-point

Returns the position where input will be taken into the input port (after the next time return is typed).

```
- (send a-text:ports get-unread-start-point) ⇒ exact-integer
```

get-value-port

Returns an output port that writes into this editor. The only difference between this port and the ports returned by `get-err-port` and `get-out-port` is the font style and color.

```
- (send a-text:ports get-value-port) ⇒ output-port
```

`get-value-style-delta`

The result of this method is the style that is used to color text submitted to the result of `get-value-port`.

This method is called during the initialization of the class.

```
- (send a-text:ports get-value-style-delta) ⇒ (is-a?/c style-delta%)  
  Returns a style delta that uses the default font (not a fixed-width font) and colors the text blue.
```

`insert-before`

Inserts some text before the unread start point and updates it and the insertion point properly. To insert between the two points, see `insert-between`.

See also `set-unread-start-point` and `set-insertion-point`.

```
- (send a-text:ports insert-before str) ⇒ void  
  str: (union snip% string)
```

`insert-between`

Inserts some text between the unread start point and the insertion point (and updates them properly). To insert before the two points, see `insert-before`.

See also `set-unread-start-point` and `set-insertion-point`.

```
- (send a-text:ports insert-between str) ⇒ void  
  str: (union snip% string)
```

`on-submit`

This method is called when text is sent into the input port.

```
- (send a-text:ports on-submit) ⇒ void  
  Does nothing.
```

`reset-input-box`

This method removes the current input box from the editor (and all input in it is lost).

```
- (send a-text:ports reset-input-box) ⇒ void
```

```
send-eof-to-box-in-port
```

This method puts an eof into the box input port.

```
- (send a-text:ports send-eof-to-box-in-port) ⇒ void
```

```
send-eof-to-in-port
```

This method puts an eof into the input port.

```
- (send a-text:ports send-eof-to-in-port) ⇒ void
```

```
set-allow-edits
```

Enables or disables editing in the buffer. Be sure to update the unread start point (via [set-unread-start-point](#)) and the insertion point (via [set-insertion-point](#)) after making changes to the buffer.

```
- (send a-text:ports set-allow-edits allow-edits?) ⇒ void
  allow-edits?: boolean
```

```
set-insertion-point
```

Sets the position where the output port will insert characters. See also [get-insertion-point](#).

```
- (send a-text:ports set-insertion-point ip) ⇒ void
  ip: exact-integer
```

```
set-unread-start-point
```

Sets the position where input will be taken into the input port (after the next time return is typed).

See also [get-unread-start-point](#).

```
- (send a-text:ports set-unread-start-point usp) ⇒ void
  usp: exact-integer
```

```
submit-to-port?
```

Augment this method to help control when characters should be submitted to the input port.

```
- (send a-text:ports submit-to-port? key) => boolean
  key: char
  Return #t or the result of calling inner.
```

25.26 `text:ports-mixin`

Domain: `text:wide-snip<%>`

Implements: `text:wide-snip<%>`

Implements: `text:ports<%>`

`can-delete?` (*augments, and augmentable only*)

Called before a range is deleted from the editor. If the return value is #f, then the delete will be aborted.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-delete` to modify the editor, if necessary.

See also `on-delete`, `after-delete`, and `on-edit-sequence`.

```
- (send a-text:ports-mixin can-delete? start len boolean
  start: exact-integer
  len: exact-integer
  Returns the results of the inner call, unless get-allow-edits returns #f.
```

`can-insert?` (*augments, and augmentable only*)

Called before `items` are inserted into the editor. If the return value is #f, then the insert will be aborted.

The editor is internally locked for writing during a call to this method (see also “Locks” (§8.8 in *PLT MrEd: Graphical Toolbox Manual*)). Use `after-insert` to modify the editor, if necessary.

See also `on-insert`, `after-insert`, and `on-edit-sequence`.

```
- (send a-text:ports-mixin can-insert? start len boolean
  start: exact-integer
  len: exact-integer
  Returns the results of the inner call, unless get-allow-edits returns #f.
```

on-display-size (*augments, and augmentable only*)

This method is called by the editor's **display** whenever the display's size (as reported by `get-view-size`) changes, but it is called indirectly through `on-display-size-when-ready`.

- (send *a-text:ports-mixin* on-display-size void
Adjusts the embedded editor-snip (used for reading input to the `get-in-box-ports`) to match the width of the editor.

on-local-char

Called by `on-char` when the event is *not* handled by a caret-owning snip.

Consider overriding `on-default-char` instead of this method.

- (send *a-text:ports-mixin* on-local-char *event* void
event: (is-a?/c `key-event%`)
Sends the data between the last position and the result of `get-unread-start-point` to the input port, unless `submit-to-port?` returns #f.
Also calls `on-submit`.

25.27 text:input-box<%>

Extends: (class->interface `text%`)

Classes that implement this interface are used as the editors for the box input port in

25.28 text:ports%

.

25.29 text:input-box-mixin

Domain: (class->interface `text%`)

Implements: `text:input-box<%>`

This mixin provides an implementation of `text:input-box<%>` for use with `text:ports<%>`.

```
- init args: [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]
  line-spacing=1.0: non-negative real number
  tab-stops=null: list of real numbers
  auto-wrap=#f: boolean
```

The `line-spacing` argument sets the additional amount of space (in DC units) inserted between each line in the editor when the editor is displayed. This spacing is included in the reported height of each line.

See `set-tabs` for information about `tabstops`.

If `auto-wrap` is true, then auto-wrapping is enabled via `auto-wrap`.

A new `keymap%` object is created for the new editor. See also `get-keymap` and `set-keymap`.

A new `style-list%` object is created for the new editor. See also `get-style-list` and `set-style-list`.

on-default-char

Called by `on-local-char` when the event is *not* handled by a caret-owning snip or by the keymap.

```
- (send a-text:input-box-mixin on-default-char event void
  event:
    key-event% object
```

Notifies the `text:ports<%>` enclosing this editor that a new line of input has been provided.

25.30 `text:basic%` = (editor:basic-mixin (text:basic-mixin text%))

```
text:basic% = (editor:basic-mixin (text:basic-mixin text%))
```

```
- (new text:basic% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) => text:basic%
  object
  line-spacing=1.0: non-negative real number
  tab-stops=null: list of real numbers
  auto-wrap=#f: boolean
```

Passes all arguments to `super-init`.

25.31 `text:hide-caret/selection%` = (text:hide-caret/selection-mixin text:basic%)

```
text:hide-caret/selection% = (text:hide-caret/selection-mixin text:basic%)
```

```
- (new text:hide-caret/selection% [(line-spacing -)] [(tab-stops -)] [(auto-wrap
-)] => text:hide-caret/selection% object
  line-spacing=1.0: non-negative real number
  tab-stops=null: list of real numbers
  auto-wrap=#f: boolean
```

Passes all arguments to `super-init`.

25.32 text:nbsp->space% = (text:nbsp->space-mixin text:basic%)

```
text:nbsp->space% = (text:nbsp->space-mixin text:basic%)
```

```
- (new text:nbsp->space% [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]) =>
  text:nbsp->space% object
  line-spacing = 1.0 : non-negative real number
  tab-stops = null : list of real numbers
  auto-wrap = #f : boolean
```

Passes all arguments to super-init.

25.33 text:delegate% = (text:delegate-mixin text:basic%)

```
text:delegate% = (text:delegate-mixin text:basic%)
```

```
- (new text:delegate% [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]) => text:delegate%
  object
  line-spacing = 1.0 : non-negative real number
  tab-stops = null : list of real numbers
  auto-wrap = #f : boolean
```

Passes all arguments to super-init.

25.34 text:wide-snip% = (text:wide-snip-mixin text:basic%)

```
text:wide-snip% = (text:wide-snip-mixin text:basic%)
```

```
- (new text:wide-snip% [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]) => text:wide-snip%
  object
  line-spacing = 1.0 : non-negative real number
  tab-stops = null : list of real numbers
  auto-wrap = #f : boolean
```

Passes all arguments to super-init.

25.35 text:standard-style-list% = (editor:standard-style-list-mixin text:wide-snip%)

```
text:standard-style-list% = (editor:standard-style-list-mixin text:wide-snip%)
```

```
- (new text:standard-style-list% [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]) =>
  text:standard-style-list% object
  line-spacing = 1.0 : non-negative real number
  tab-stops = null : list of real numbers
  auto-wrap = #f : boolean
```

Passes all arguments to super-init.

25.36 text:input-box% = (text:input-box-mixin text:standard-style-list%)

```
text:input-box% = (text:input-box-mixin text:standard-style-list%)
```

```
- (new text:input-box% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) => text:input-bo
object
```

```
  line-spacing = 1.0 : non-negative real number
```

```
  tab-stops = null : list of real numbers
```

```
  auto-wrap = #f : boolean
```

Passes all arguments to `super-init`.

25.37 `text:keymap%` = (`editor:keymap-mixin` `text:standard-style-list%`)

```
text:keymap% = (editor:keymap-mixin text:standard-style-list%)
```

```
- (new text:keymap% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) => text:keymap%
object
```

```
  line-spacing = 1.0 : non-negative real number
```

```
  tab-stops = null : list of real numbers
```

```
  auto-wrap = #f : boolean
```

Passes all arguments to `super-init`.

25.38 `text:return%` = (`text:return-mixin` `text:keymap%`)

```
text:return% = (text:return-mixin text:keymap%)
```

```
- (new text:return% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)] (return
-)) => text:return% object
```

```
  line-spacing = 1.0 : non-negative real number
```

```
  tab-stops = null : list of real numbers
```

```
  auto-wrap = #f : boolean
```

```
  return : (-i boolean)
```

Passes all arguments to `super-init`.

25.39 `text:autowrap%` = (`editor:autowrap-mixin` `text:keymap%`)

```
text:autowrap% = (editor:autowrap-mixin text:keymap%)
```

```
- (new text:autowrap% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) => text:autowrap%
object
```

```
  line-spacing = 1.0 : non-negative real number
```

```
  tab-stops = null : list of real numbers
```

```
  auto-wrap = #f : boolean
```

Passes all arguments to `super-init`.

25.40 `text:file%` = (`text:file-mixin` (`editor:file-mixin` `text:autowrap%`))

25. Text 25.41. text:clever-file-format% = (text:clever-file-format-mixin text:file%)

```
text:file% = (text:file-mixin (editor:file-mixin text:autowrap%))
```

- (new text:file% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) ⇒ text:file% object
 - line-spacing = 1.0 : non-negative real number
 - tab-stops = null : list of real numbers
 - auto-wrap = #f : boolean

Passes all arguments to super-init.

25.41 text:clever-file-format% = (text:clever-file-format-mixin text:file%)

```
text:clever-file-format% = (text:clever-file-format-mixin text:file%)
```

- (new text:clever-file-format% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) ⇒ text:clever-file-format% object
 - line-spacing = 1.0 : non-negative real number
 - tab-stops = null : list of real numbers
 - auto-wrap = #f : boolean

Passes all arguments to super-init.

25.42 text:backup-autosave% = (editor:backup-autosave-mixin text:clever-file-format%)

```
text:backup-autosave% = (editor:backup-autosave-mixin text:clever-file-format%)
```

- (new text:backup-autosave% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) ⇒ text:backup-autosave% object
 - line-spacing = 1.0 : non-negative real number
 - tab-stops = null : list of real numbers
 - auto-wrap = #f : boolean

Passes all arguments to super-init.

25.43 text:searching% = (text:searching-mixin text:backup-autosave%)

```
text:searching% = (text:searching-mixin text:backup-autosave%)
```

- (new text:searching% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) ⇒ text:searching% object
 - line-spacing = 1.0 : non-negative real number
 - tab-stops = null : list of real numbers
 - auto-wrap = #f : boolean

Passes all arguments to super-init.

25.44 `text:info% = (editor:info-mixin (text:info-mixin text:searching%))`

`text:info% = (editor:info-mixin (text:info-mixin text:searching%))`

- (new text:info% [(line-spacing -)] [(tab-stops -)] [(auto-wrap -)]) ⇒ `text:info%` object
 - `line-spacing = 1.0`: non-negative real number
 - `tab-stops = null`: list of real numbers
 - `auto-wrap = #f`: boolean

Passes all arguments to `super-init`.

26. Version

27. Framework Functions

27.1 Framework Functions

`application:current-app-name`

- `(application:current-app-name) ⇒ string?`
- `(application:current-app-name name) ⇒ void?`
`name: string?`

This is a parameter specifying the name of the current application. It is used in the help menu (see `frame:standard-menus%`) and in frame titles (see `frame:editor%`).

The first case in the case-lambda returns the current name, and the second case in the case-lambda sets the name of the application to `name`.

`autosave:register`

- `(autosave:register obj) ⇒ void?`
`obj: (and/c (is-a?/c autosave:autosavable<%>) (is-a?/c editor<%>))`

Adds `obj` to the list of objects to be autosaved. When it is time to autosave, the `do-autosave` method of the object is called. This method is responsible for performing the autosave.

There is no need to de-register an object because the autosaver keeps a “weak” pointer to the object; i.e., the autosaver does not keep an object from garbage collection.

`autosave:restore-autosave-files/gui`

- `(autosave:restore-autosave-files/gui) ⇒ void?`

Opens a GUI to ask the user about recovering any autosave files left around from crashes and things.

This function doesn't return until the user has finished restoring the autosave files. (It uses `yield` to handle events however).

`color-model:rgb->xyz`

- (`color-model:rgb->xyz r g b`) \Rightarrow `color-model:xyz?`
`r`: number?
`g`: number?
`b`: number?

Converts a color represented as a red-green-blue tuple (each value from 0 to 255) into an XYZ tuple. This describes a point in the CIE XYZ color space.

`color-model:rgb-color-distance`

- (`color-model:rgb-color-distance red-a green-a blue-a red-b green-b blue-b`) \Rightarrow `number?`
`red-a`: number?
`green-a`: number?
`blue-a`: number?
`red-b`: number?
`green-b`: number?
`blue-b`: number?

This calculates a distance between two colors. The smaller the distance, the closer the colors should appear to the human eye. A distance of 10 is reasonably close that it could be called the same color.

This function is not symmetric in red, green, and blue, so it is important to pass red, green, and blue components of the colors in the the proper order. The first three arguments are red, green and blue for the first color, respectively, and the second three arguments are red green and blue for the second color, respectively.

`color-model:xyz->rgb`

- (`color-model:xyz->rgb x y z`) \Rightarrow (list/c number? number? number?)
`x`: number?
`y`: number?
`z`: number?

Converts an XYZ-tuple (in the CIE XYZ colorspace) into a list of values representing an RGB-tuple.

`color-model:xyz-x`

- (`color-model:xyz-x xyz`) \Rightarrow `number?`
`xyz`: `color-model:xyz?`

Extracts the x component of `xyz`.

`color-model:xyz-y`

- (`color-model:xyz-y xyz`) \Rightarrow `number?`
`xyz` : `color-model:xyz?`

Extracts the y component of `xyz`.

`color-model:xyz-z`

- (`color-model:xyz-z xyz`) \Rightarrow `number?`
`xyz` : `color-model:xyz?`

Extracts the z component of `xyz`.

`color-model:xyz?`

- (`color-model:xyz? val`) \Rightarrow `boolean?`
`val` : `any/c`

Determines if `val` an `xyz` color record.

`color-prefs:add-background-preferences-panel`

- (`color-prefs:add-background-preferences-panel`) \Rightarrow `void?`
 Adds a preferences panel that configures the background color for `editor:basic-mixin`.

`color-prefs:add-to-preferences-panel`

- (`color-prefs:add-to-preferences-panel name func`) \Rightarrow `void?`
`name` : `string?`
`func` : ((`is-a?/c vertical-panel%`) . -> . `void?`)

Calls `func` with the subpanel of the preferences coloring panel that corresponds to `name`.

`color-prefs:build-color-selection-panel`

- (`color-prefs:build-color-selection-panel parent pref-sym style-name example-text`)
 \Rightarrow `void?`
`parent` : (`is-a?/c area-container<%>`)

```

pref-sym : symbol?
style-name : string?
example-text : string?

```

Builds a panel with a number of controls for configuring a font: the color and check boxes for bold, italic, and underline. The *parent* argument specifies where the panel will be placed. The *pref-sym* should be a preference (suitable for use with `preferences:get` and `preferences:set`). The *style-name* specifies the name of a style in the style list returned from `editor:get-standard-style-list` and *example-text* is shown in the panel so users can see the results of their configuration.

```
color-prefs:marshall-style
```

```
- (color-prefs:marshall-style style-delta) ⇒ printable/c
  style-delta : (is-a?/c style-delta%)
```

Builds a printed representation for a style-delta.

```
color-prefs:register-color-pref
```

```
- (color-prefs:register-color-pref pref-name style-name color) ⇒ void?
  pref-name : symbol?
  style-name : string?
  color : (is-a?/c color%)
```

This function registers a color preference and initializes the style list returned from `editor:get-standard-style-list`. In particular, it calls `preferences:set-default` and `preferences:set-un/marshall` to install the pref for *pref-name*, using *color* as the default color. The preference is bound to a *style-delta*%, and initially the *style-delta*% changes the foreground color to *color*. Then, it calls `editor:set-standard-style-list-delta` passing the *style-name* and the current value of the preference *pref-name*.

Finally, it adds calls `preferences:add-callback` to set a callback for *pref-name* that updates the style list when the preference changes.

```
color-prefs:unmarshall-style
```

```
- (color-prefs:unmarshall-style marshalled-style-delta) ⇒ (union false/c (is-a?/c style-delta%)
  marshalled-style-delta : printable/c
```

Builds a style delta from its printed representation. Returns #f if the printed form cannot be parsed.

```
editor:get-default-color-style-name
```

```
- (editor:get-default-color-style-name) ⇒ string?
```

The name of the style (in the list returned by `editor:get-standard-style-list`) that holds the default color.

`editor:get-standard-style-list`

- (`editor:get-standard-style-list`) ⇒ (is-a?/c `style-list%`)

Returns a style list that is used for all instances of `editor:standard-style-list<%>`.

`editor:set-default-font-color`

- (`editor:set-default-font-color color`) ⇒ void?
`color`: (is-a?/c `color%`)

Sets the color of the style named `editor:get-default-color-style-name`.

`editor:set-standard-style-list-delta`

- (`editor:set-standard-style-list-delta name delta`) ⇒ void?
`name`: string?
`delta`: (is-a?/c `style-delta%`)

Finds (or creates) the style named by `name` in the result of `editor:get-standard-style-list` and sets its delta to `delta`.

If the style named by `name` is already in the style list, it must be a delta style.

`editor:set-standard-style-list-pref-callbacks`

- (`editor:set-standard-style-list-pref-callbacks`) ⇒ any

Installs the font preference callbacks that update the style list returned by `editor:get-standard-style-list` based on the font preference symbols.

`exit:can-exit?`

- (`exit:can-exit?`) ⇒ boolean?

Calls the “can-callbacks” and returns their results. See `exit:insert-can?-callback` for more information.

`exit:exit`

- (`exit:exit`) ⇒ any

`exit:exit` performs four actions:

- sets the result of the `exit:exiting?` function to `#t`.
- invokes the exit-callbacks, with `exit:can-exit?` If none of the “can?” callbacks return `#f`,
- invokes `exit:on-exit` and then
- queues a callback that calls `exit` (a mzscheme procedure) and (if `exit` returns) sets the result of `exit:exiting?` back to `#t`.

`exit:exiting?`

- `(exit:exiting?) ⇒ boolean?`

Returns `#t` to indicate that an exit operation is taking place. Does not indicate that the app will actually exit, since the user may cancel the exit.

See also `exit:insert-on-callback` and `exit:insert-can?-callback`.

`exit:insert-can?-callback`

- `(exit:insert-can?-callback callback) ⇒ (-> void?)`
`callback: (-> boolean?)`

Use this function to add a callback that determines if an attempted exit can proceed. This callback should not clean up any state, since another callback may veto the exit. Use `exit:insert-on-callback` for callbacks that clean up state.

`exit:insert-on-callback`

- `(exit:insert-on-callback callback) ⇒ (-> void?)`
`callback: (-> void?)`

Adds a callback to be called when exiting. This callback must not fail. If a callback should stop an exit from happening, use `exit:insert-can?-callback`.

`exit:on-exit`

- `(exit:on-exit) ⇒ void?`

Calls the “on-callbacks”. See `exit:insert-on-callback` for more information.

`exit:set-exiting`

- `(exit:set-exiting exiting?) ⇒ void?`
`exiting?: boolean?`

Sets a flag that affects the result of `exit:exiting?`.

`exit:user-oks-exit`

- `(exit:user-oks-exit) ⇒ boolean?`

Opens a dialog that queries the user about exiting. Returns the user's decision.

`exn:exn?`

- `(exn:exn? exn) ⇒ boolean?`
`exn: any/c`

Tests if a value is a framework exception.

`exn:make-exn`

- `(exn:make-exn message continuation-marks) ⇒ exn?`
`message: string?`
`continuation-marks: continuation-mark-set?`

Creates a framework exception.

`exn:make-unknown-preference`

- `(exn:make-unknown-preference message continuation-marks) ⇒ exn:unknown-preference?`
`message: string?`
`continuation-marks: continuation-mark-set?`

Creates an unknown preference exception.

`exn:unknown-preference?`

- `(exn:unknown-preference? exn) ⇒ boolean?`
`exn: any/c`

Determines if a value is an unknown preference exn.

`finder:common-get-file`

- `(finder:common-get-file directory prompt filter filter-msg parent) ⇒ (union path? false/c)`

```

directory = #f : (union path? false/c)
prompt = "Select File" : string?
filter = #f : (union byte-regexp? false/c)
filter-msg = "That filename does not have the right form." : string?
parent = #f : (union false/c (is-a?/c top-level-window<%>))

```

This procedure queries the user for a single filename, using a platform-independent dialog box. Consider using `finder:get-file` instead of this function.

See section 11 for more information.

`finder:common-get-file-list`

```

- (finder:common-get-file-list directory prompt filter filter-msg parent) ⇒
(union (listof path?) false/c)
  directory = #f : (union false/c path?)
  prompt = "Select File" : string?
  filter = #f : (union false/c byte-regexp?)
  filter-msg = "That filename does not have the right form." : string?
  parent = #f : (union false/c (is-a?/c top-level-window<%>))

```

This procedure queries the user for a list of filenames, using a platform-independent dialog box.

See section 11 for more information.

`finder:common-put-file`

```

- (finder:common-put-file name directory replace? prompt filter filter-msg parent)
⇒ (union false/c path?)
  name = "Untitled" : string?
  directory = #f : (union false/c path?)
  replace? = #f : boolean?
  prompt = "Select File" : string?
  filter = #f : (union false/c byte-regexp?)
  filter-msg = "That filename does not have the right form." : string?
  parent = (finder:dialog-parent-parameter) : (union (is-a?/c top-level-window<%>) false/c)

```

This procedure queries the user for a single filename, using a platform-independent dialog box. Consider using `finder:put-file` instead of this function.

See section 11 for more information.

`finder:default-extension`

```

- (finder:default-extension) ⇒ string?
- (finder:default-extension extension) ⇒ void?
  extension : string?

```

This parameter controls the default extension for the framework's `finder:put-file` dialog. Its value gets passed as the `default-extension` argument to `put-file`.

Its default value is "".

`finder:default-filters`

- (`finder:default-filters filters`) ⇒ `void?`
`filters`: (`listof (list/c string? string?)`)
- (`finder:default-filters`) ⇒ (`listof (list/c string? string?)`)

This parameter controls the default extension for the framework's `finder:put-file` dialog. Its value gets passed as the `default-filters` argument to `put-file`.

Its default value is ' ("Any" "*. *")).

`finder:get-file`

- (`finder:get-file directory prompt filter filter-msg parent`) ⇒ (`union path? false/c`)
`directory` = #f: (`union path? false/c`)
`prompt` = "Select File": `string?`
`filter` = #f: (`union byte-regexp? string? false/c`)
`filter-msg` = "That filename does not have the right form.": `string?`
`parent` = #f: (`union false/c (is-a?/c top-level-window<%>)`)

Queries the user for a filename.

If the result of (`preferences:get 'framework:file-dialogs`) is 'std this calls `finder:std-get-file`, and if it is 'common, `finder:common-get-file` is called.

`finder:put-file`

- (`finder:put-file name directory replace? prompt filter filter-msg parent`)
⇒ (`union false/c path?`)
`name` = "Untitled": `string?`
`directory` = #f: (`union false/c path?`)
`replace?` = #f: `boolean?`
`prompt` = "Select File": `string?`
`filter` = #f: (`union false/c byte-regexp?`)
`filter-msg` = "That filename does not have the right form.": `string?`
`parent` = (`finder:dialog-parent-parameter`): (`union (is-a?/c top-level-window<%>) false/c`)

Queries the user for a filename.

If the result of (`preferences:get 'framework:file-dialogs`) is 'std this calls `finder:std-put-file`, and if it is 'common, `finder:common-put-file` is called.

finder:std-get-file

- (finder:std-get-file *directory prompt filter filter-msg parent*) ⇒ (union path? false/c)
 - directory* = #f: (union path? false/c)
 - prompt* = "Select File": string?
 - filter* = #f: (union byte-regexp? false/c)
 - filter-msg* = "That filename does not have the right form.": string?
 - parent* = #f: (union false/c (is-a?/c top-level-window<%>))

This procedure queries the user for a single filename, using a platform-dependent dialog box. Consider using `finder:get-file` instead of this function.

See section 11 for more information.

finder:std-put-file

- (finder:std-put-file *name directory replace? prompt filter filter-msg parent*) ⇒ (union false/c path?)
 - name* = "Untitled": string?
 - directory* = #f: (union false/c path?)
 - replace?* = #f: boolean?
 - prompt* = "Select File": string?
 - filter* = #f: (union false/c byte-regexp?)
 - filter-msg* = "That filename does not have the right form.": string?
 - parent* = (finder:dialog-parent-parameter): (union (is-a?/c top-level-window<%>) false/c)

This procedure queries the user for a single filename, using a platform-dependent dialog box. Consider using `finder:put-file` instead of this function.

See section 11 for more information.

frame:add-snip-menu-items

- (frame:add-snip-menu-items *menu menu-item%*) ⇒ void?
 - menu*: (is-a?/c menu%)
 - menu-item%*: (subclass?/c menu-item%)

Inserts three menu items into *menu*, one that inserts a text box, one that inserts a pasteboard box, and one that inserts an image into the currently focused editor (if there is one). Uses *menu-item%* as the class for the menu items.

frame:reorder-menus

- (frame:reorder-menus *frame*) ⇒ void?
 - frame*: (is-a?/c frame%)

Re-orders the menus in a frame. It moves the “File” and “Edit” menus to the front of the menubar and moves the “Windows” and “Help” menus to the end of the menubar.

This is useful in conjunction with the frame classes. After instantiating the class and adding ones own menus, the menus will be mis-ordered. This function fixes them up.

`frame:setup-size-pref`

- (`frame:setup-size-pref` *size-pref-sym* *width* *height*) ⇒ `void`
size-pref-sym: `symbol?`
width: `number?`
height: `number?`

Initializes a preference for the `frame:size-pref` mixin.

The first argument should be the preferences symbol, and the second and third should be the default width and height, respectively.

`group:get-the-frame-group`

- (`group:get-the-frame-group`) ⇒ (`is-a?/c` `group:%`)

This returns the frame group.

`gui-utils:cancel-on-right?`

- (`gui-utils:cancel-on-right?`) ⇒ `boolean?`

Returns `#t` if cancel should be on the right-hand side (or below) in a dialog and `#f` otherwise.

See also `gui-utils:ok/cancel-buttons`.

`gui-utils:cursor-delay`

- (`gui-utils:cursor-delay`) ⇒ `real?`
- (`gui-utils:cursor-delay` *new-delay*) ⇒ `void?`
new-delay: `real?`

This function is *not* a parameter. Instead, the state is just stored in the closure.

The first case in the case lambda returns the current delay in seconds before a watch cursor is shown, when either `gui-utils:local-busy-cursor` or `gui-utils:show-busy-cursor` is called. The second case in the case lambda Sets the delay, in seconds, before a watch cursor is shown, when either `gui-utils:local-busy-cursor` or `gui-utils:show-busy-cursor` is called.

gui-utils:delay-action

```
- (gui-utils:delay-action delay-time open close) ⇒ void?
  delay-time: real?
  open: (-> void?)
  close: (-> void?)
```

Use this function to delay an action for some period of time. It also supports cancelling the action before the time period elapses. For example, if you want to display a watch cursor, but you only want it to appear after 2 seconds and the action may or may not take more than two seconds, use this pattern:

```
(let ([close-down
      (gui-utils:delay-action
       2
       ( () .. init watch cursor ...)
       ( () .. close watch cursor ...))])
  ;; .. do action ...
  (close-down))
```

Creates a thread that waits *delay-time*. After *delay-time* has elapsed, if the result thunk has *not* been called, call *open*. Then, when the result thunk is called, call *close*. The function *close* will only be called if *open* has been called.

gui-utils:get-choice

```
- (gui-utils:get-choice message true-choice false-choice title default-result
  paren style) ⇒ any/c
  message: string?
  true-choice: string?
  false-choice: string?
  title = (string-constant warning): string?
  default-result = (quote disallow-close): any/c
  paren = #f: (union false/c (is-a?/c frame%) (is-a?/c dialog%))
  style = (quote app): (symbols 'app 'caution 'stop)
```

Opens a dialog that presents a binary choice to the user. The user is forced to choose between these two options, ie cancelling or closing the dialog opens a message box asking the user to actually choose one of the two options.

The dialog will contain the string *message* and two buttons, labeled with the *true-choice* and the *false-choice*. If the user clicks on *true-choice* #t is returned. If the user clicks on *false-choice*, #f is returned.

The argument *default-result* determines how closing the window is treated. If the argument is 'disallow-close, closing the window is not allowed. If it is anything else, that value is returned when the user closes the window.

If **gui-utils:cancel-on-right?** returns #t, the false choice is on the right. Otherwise, the true choice is on the right.

The *style* parameter is (eventually) passed to **message%** as an icon in the dialog.

`gui-utils:get-clickback-delta`

- `(gui-utils:get-clickback-delta) ⇒ (is-a?/c style-delta%)`

This delta is designed for use with `set-clickback`. Use the result of this function as the style for the region text where the clickback is set.

See also `gui-utils:get-clicked-clickback-delta`.

`gui-utils:get-clicked-clickback-delta`

- `(gui-utils:get-clicked-clickback-delta) ⇒ (is-a?/c style-delta%)`

This delta is designed for use with `set-clickback`. Use it as one of the `style-delta%` argument to `set-clickback`.

See also `gui-utils:get-clickback-delta`.

`gui-utils:local-busy-cursor`

- `(gui-utils:local-busy-cursor window thunk delay) ⇒ any/c`
`window : (is-a?/c window<*>)`
`thunk : (-> any/c)`
`delay = (gui-utils:cursor-delay) : integer?`

Evaluates `(thunk)` with a watch cursor in `window`. If `window` is `#f`, the watch cursor is turned on globally. The argument `delay` specifies the amount of time before the watch cursor is opened. Use `gui-utils:cursor-delay` to set this value for all uses of this function.

The result of this function is the result of `thunk`.

`gui-utils:next-untitled-name`

- `(gui-utils:next-untitled-name) ⇒ string?`

Returns a name for the next opened untitled frame. The first name is “Untitled”, the second is “Untitled 2”, the third is “Untitled 3”, and so forth.

`gui-utils:ok/cancel-buttons`

- `(gui-utils:ok/cancel-buttons parent confirm-callback cancel-callback confirm-label cancel-label) ⇒ (values (is-a?/c button%) (is-a?/c button%))`
`parent : (is-a?/c area-container<*>)`
`confirm-callback : ((is-a?/c button%) (is-a?/c event%)) .-> . any`

```
cancel-callback: ((is-a?/c button%) (is-a?/c event%)) .-> . any)
confirm-label = (string-constant ok) : string?
cancel-label = (string-constant cancel) : string?
```

Adds an Ok and a cancel button to a panel, changing the order to suit the platform. Under Mac OS X and unix, the confirmation action is on the right (or bottom) and under Windows, the canceling action is on the right (or bottom). The confirmation action button has the ' (border) style. The buttons are also sized to be the same width.

The first result is be the OK button and the second is the cancel button.

See also [gui-utils:cancel-on-right?](#).

gui-utils:show-busy-cursor

```
- (gui-utils:show-busy-cursor thunk delay) => any/c
  thunk : (-> any/c)
  delay = (gui-utils:cursor-delay) : integer?
```

Evaluates (*thunk*) with a watch cursor. The argument *delay* specifies the amount of time before the watch cursor is opened. Use [gui-utils:cursor-delay](#) to set this value to all calls.

This function returns the result of *thunk*.

gui-utils:trim-string

```
- (gui-utils:trim-string str size) => (and/c string? ((str) (<= (string-length str) size)))
  str : string?
  size : (and/c number? positive?)
```

Constructs a string whose size is less than *size* by trimming the *str* and inserting an ellipses into it.

gui-utils:unsaved-warning

```
- (gui-utils:unsaved-warning filename action can-save-now? parent) => (symbols
'continue 'save 'cancel)
  filename : string?
  action : string?
  can-save-now? = #f : boolean?
  parent = #f : (union false/c (is-a?/c frame%) (is-a?/c dialog%))
```

This displays a dialog that warns the user of a unsaved file.

The string, *action*, indicates what action is about to take place, without saving. For example, if the application is about to close a file, a good action is "Close Anyway". The result symbol indicates the user's choice. If *can-save-now?* is #f, this function does not give the user the "Save" option and thus will not return 'save.

handler:add-to-recent

- (handler:add-to-recent *filename*) ⇒ void?
filename: path?

Adds a filename to the list of recently opened files.

handler:current-create-new-window

- (handler:current-create-new-window *new-window-handler*) ⇒ void
new-window-handler: ((union false/c path?) .-> . (is-a?/c frame%))
- (handler:current-create-new-window) ⇒ ((union false/c string?) .-> . (is-a?/c frame%))

This is a parameter that controls how the framework creates new application windows.

The default setting is this:

```
( (filename)
  (let ([frame (make-object frame:text-info-file% filename)]
        (send frame show #t)
        frame))
```

handler:edit-file

- (handler:edit-file *filename* *make-default*) ⇒ (union false/c (is-a?/c frame:editor<%>))
filename: (union path? false/c)
make-default = (() ((handler:current-create-new-window) *filename*)) : (-> (is-a?/c frame:editor<%>))

This function creates a frame or re-uses an existing frame to edit a file.

If the preference 'framework:open-here is set to #t, and (send ([group:get-the-frame-group](#)) [get-open-here-frame](#)) returns a frame, the [open-here](#) method of that frame is used to load the file in the existing frame.

Otherwise, it invokes the appropriate format handler to open the file (see [handler:insert-format-handler](#)).

- If *filename* is a string, this function checks the result of [group:get-the-frame-group](#) to see if the *filename* is already open by a frame in the group.
 - * If so, it returns the frame.
 - * If not, this function calls [handler:find-format-handler](#) with *filename*.
 - If a handler is found, it is applied to *filename* and it's result is the final result.
 - If not, *make-default* is used.
- If *filename* is #f, *make-default* is used.

handler:find-format-handler

- (handler:find-format-handler *filename*) ⇒ (path? .-> . (is-a?/c frame:editor<%>))
filename: path?

This function selects a format handler. See also [handler:insert-format-handler](#).

It finds a handler based on *filename*.

`handler:find-named-format-handler`

- (`handler:find-named-format-handler name`) ⇒ (path? .-> . (is-a?/c `frame:editor<%>`))
`name` : string?

This function selects a format handler. See also `handler:insert-format-handler`.

It finds a handler based on `name`.

`handler:handler-extension`

- (`handler:handler-extension handler`) ⇒ (union (path? .-> . boolean?) (listof string?))
`handler` : handler:handler?

Extracts the extension from a handler.

`handler:handler-handler`

- (`handler:handler-handler handler`) ⇒ (path? .-> . (is-a?/c `frame:editor<%>`))
`handler` : handler:handler?

Extracts the handler's handling function

`handler:handler-name`

- (`handler:handler-name handler`) ⇒ string?
`handler` : handler:handler?

Extracts the name from a handler.

`handler:handler?`

- (`handler:handler? obj`) ⇒ boolean?
`obj` : any/c

This predicate determines if its input is a handler

`handler:insert-format-handler`

- (`handler:insert-format-handler name pred handler`) ⇒ void?
`name` : string?

```

pred : (union string? (listof string?) (path? . -> . boolean?))
handler : (path? . -> . (union false/c (is-a?/c frame:editor<%>)))

```

This function inserts a format handler.

The string, *name* names the format handler for use with `handler:find-named-format-handler`. If *pred* is a string, it is matched with the extension of a filename by `handler:find-format-handler`. If *pred* is a list of strings, they are each matched with the extension of a filename by `handler:find-format-handler`. If it is a function, the filename is applied to the function and the functions result determines if this is the handler to use.

The most recently added format handler takes precedence over all other format handlers.

handler:install-recent-items

```

- (handler:install-recent-items menu) ⇒ void?
  menu : (is-a?/c menu%)

```

This function deletes all of the items in the given menu and adds one menu item for each recently opened file. These menu items, when selected, call `handler:edit-file` with the filename of the recently opened file.

The menu's size is limited to 10.

handler:open-file

```

- (handler:open-file) ⇒ (union false/c (is-a?/c frame:basic<%>))

```

This function queries the user for a filename and opens the file for editing. It uses `handler:edit-file` to open the file, once the user has chosen it.

Calls `finder:get-file` and `handler:edit-file`.

handler:set-recent-items-frame-superclass

```

- (handler:set-recent-items-frame-superclass frame) ⇒ void?
  frame : (implementation?/c frame:standard-menus<%>)

```

Sets the superclass for the recently opened files frame. It must be derived from `frame:standard-menus<%>`.

handler:set-recent-position

```

- (handler:set-recent-position filename start end) ⇒ void?
  filename : string?
  start : number?
  end : number?

```

Sets the selection of the recently opened file to *start* and *end*.

handler:size-recently-opened-files

- (handler:size-recently-opened-files *num*) ⇒ void?
num: number?
 Sizes the 'framework:recently-opened-files/pos preference list length to *num*.

icon:get-anchor-bitmap

- (icon:get-anchor-bitmap) ⇒ (is-a?/c *bitmap%*)
 This returns the anchor's *bitmap%*.
 The bitmap may not respond \#t to the *ok?* method.

icon:get-autowrap-bitmap

- (icon:get-autowrap-bitmap) ⇒ (is-a?/c *bitmap%*)
 This returns the autowrap's *bitmap%*.
 The bitmap may not respond \#t to the *ok?* method.

icon:get-eof-bitmap

- (icon:get-eof-bitmap) ⇒ (is-a?/c *bitmap%*)
 This returns the *bitmap%* used for the clickable "eof" icon from *text:ports<%>*.

icon:get-gc-off-bitmap

- (icon:get-gc-off-bitmap) ⇒ (is-a?/c *bitmap%*)
 This returns a bitmap to be displayed in an *frame:info<%>* frame when garbage collection is *not* taking place.
 The bitmap may not respond \#t to the *ok?* method.

icon:get-gc-on-bitmap

- (icon:get-gc-on-bitmap) ⇒ (is-a?/c *bitmap%*)
 This returns a bitmap to be displayed in an *frame:info<%>* frame when garbage collection is taking place.
 The bitmap may not respond \#t to the *ok?* method.

`icon:get-left/right-cursor`

- (`icon:get-left/right-cursor`) ⇒ (is-a?/c `cursor%`)

This function returns a `cursor%` object that indicates left/right sizing is possible, for use with columns inside a window.

The cursor may not respond `\#t` to the `ok?` method.

`icon:get-lock-bitmap`

- (`icon:get-lock-bitmap`) ⇒ (is-a?/c `bitmap%`)

This returns the lock's `bitmap%`.

The bitmap may not respond `\#t` to the `ok?` method.

`icon:get-paren-highlight-bitmap`

- (`icon:get-paren-highlight-bitmap`) ⇒ (is-a?/c `bitmap%`)

This returns the parenthesis highlight `bitmap%`. It is only used on black and white screens.

`icon:get-unlock-bitmap`

- (`icon:get-unlock-bitmap`) ⇒ (is-a?/c `bitmap%`)

This returns the reset unlocked `bitmap%`.

The bitmap may not respond `\#t` to the `ok?` method.

`icon:get-up/down-cursor`

- (`icon:get-up/down-cursor`) ⇒ (is-a?/c `cursor%`)

This function returns a `cursor%` object that indicates up/down sizing is possible, for use with columns inside a window.

The cursor may not respond `\#t` to the `ok?` method.

`keymap:add-to-right-button-menu`

- (`keymap:add-to-right-button-menu func`) ⇒ `void?`
`func` : ((is-a?/c `popup-menu%`) (is-a?/c `editor<%>`) (is-a?/c `event%`) . -> . `void?`)

- (`keymap:add-to-right-button-menu`) ⇒ ((`is-a?/c popup-menu%`) (`is-a?/c editor<%>`) (`is-a?/c event%`)). -> . void?

When the keymap that `keymap:get-global` returns is installed into an editor, this parameter's value is used for right button clicks.

Before calling this procedure, the function `append-editor-operation-menu-items` is called.

See also `keymap:add-to-right-button-menu/before`.

`keymap:add-to-right-button-menu/before`

- (`keymap:add-to-right-button-menu/before func`) ⇒ void?
`func`: ((`is-a?/c popup-menu%`) (`is-a?/c editor<%>`) (`is-a?/c event%`)). -> . void?
- (`keymap:add-to-right-button-menu/before`) ⇒ ((`is-a?/c popup-menu%`) (`is-a?/c editor<%>`) (`is-a?/c event%`)). -> . void?

When the keymap that `keymap:get-global` returns is installed into an editor, this function is called for right button clicks.

After calling this procedure, the function `append-editor-operation-menu-items` is called.

See also `keymap:add-to-right-button-menu`.

`keymap:add-user-keybindings-file`

- (`keymap:add-user-keybindings-file user-keybindings-path`) ⇒ any
`user-keybindings-path`: path?

Chains the keymap defined by `user-keybindings-path` to the global keymap, returned by `keymap:get-global`.

`keymap:call/text-keymap-initializer`

- (`keymap:call/text-keymap-initializer thunk-proc`) ⇒ any/c
`thunk-proc`: (-> any/c)

Thus function parameterizes the call to `thunk-proc` by setting the keymap-initialization procedure (see `current-text-keymap-initializer`) to install the framework's standard text bindings.

`keymap:canonicalize-keybinding-string`

- (`keymap:canonicalize-keybinding-string keybinding-string`) ⇒ string?
`keybinding-string`: string?

Returns a string that denotes the same keybindings as the input string, except that it is in canonical form; two canonical keybinding strings can be compared with `string=?`.

`keymap:get-editor`

- `(keymap:get-editor) ⇒ (is-a?/c keymap%)`

This returns a keymap for handling standard editing operations. It binds these keys:

- **z**: undo
- **y**: redo
- **x**: cut
- **c**: copy
- **v**: paste
- **a**: select all

where each key is prefixed with the menu-shortcut key, based on the platform. Under unix, the shortcut is `scm"a:`"; under windows the shortcut key is `"c: "` and under MacOS, the shortcut key is `"d: "`.

`keymap:get-file`

- `(keymap:get-file) ⇒ (is-a?/c keymap%)`

This returns a keymap for handling file operations.

`keymap:get-global`

- `(keymap:get-global) ⇒ (is-a?/c keymap%)`

This returns a keymap for general operations. See `keymap:setup-global` for a list of the bindings this keymap contains.

`keymap:get-search`

- `(keymap:get-search) ⇒ (is-a?/c keymap%)`

This returns a keymap for searching operations

`keymap:make-meta-prefix-list`

- `(keymap:make-meta-prefix-list key) ⇒ (listof string?)`
`key: string?`

This prefixes a key with all of the different meta prefixes and returns a list of the prefixed strings.

takes a keymap, a base key specification, and a function name; it prefixes the base key with all “meta” combination prefixes, and installs the new combinations into the keymap. For example, `(keymap:send-map-function-meta keymap "a" func)` maps all of “m:a” and “ESC;a” to `func`.

`keymap:remove-chained-keymap`

- (`keymap:remove-chained-keymap editor keymap`) ⇒ `void?`
`editor`: (`is-a?/c editor`<%>)
`keymap`: (`is-a?/c keymap:aug-keymap`<%>)

Removes `keymap` from the keymaps chained to `editor`. Also (indirectly) removes all keymaps chained to `keymap` from `editor`, since they are removed when unchaining `keymap` itself.

Each of the keymaps chained to `editor` must be an `keymap:aug-keymap`<%> and `keymap` cannot be the result of

```
(send editor get-keymap)
```

That is, `keymap` must be chained to some keymap attached to the editor.

`keymap:remove-user-keybindings-file`

- (`keymap:remove-user-keybindings-file user-keybindings-path`) ⇒ `any`
`user-keybindings-path`: `path?`

Removes the keymap previously added by `keymap:add-user-keybindings-file`.

`keymap:send-map-function-meta`

- (`keymap:send-map-function-meta keymap key func`) ⇒ `void?`
`keymap`: (`is-a?/c keymap`%)
`key`: `string?`
`func`: `string?`

Most keyboard and mouse mappings are inserted into a keymap by calling the keymap's `map-function` method. However, “meta” combinations require special attention. The “m:” prefix recognized by `map-function` applies only to the Meta key that exists on some keyboards. By convention, however, “meta” combinations can also be accessed by using “ESC” as a prefix.

This procedure binds all of the key-bindings obtained by prefixing `key` with a meta-prefix to `func` in `keymap`.

`keymap:set-chained-keymaps`

- (`keymap:set-chained-keymaps keymap children-keymaps`) ⇒ `void?`
`keymap`: (`is-a?/c keymap:aug-keymap`<%>)
`children-keymaps`: (`listof (is-a?/c keymap`%)

Sets `keymap`'s chained keymaps to `children-keymaps`, unchaining any keymaps that are currently chained to `keymap`.

keymap:setup-editor

- (keymap:setup-editor keymap) ⇒ void?
keymap: (is-a?/c keymap%)

This sets up the input keymap with the bindings described in `keymap:get-editor`.

keymap:setup-file

- (keymap:setup-file keymap) ⇒ void?
keymap: (is-a?/c keymap%)

This extends a `keymap%` with the bindings for files.

keymap:setup-global

- (keymap:setup-global keymap) ⇒ void?
keymap: (is-a?/c keymap%)

This extends a `keymap%` with the general bindings.

This function extends a `keymap%` with the following functions:

- “ring-bell” (*any events*) — Rings the bell (using `bell`) and removes the search panel from the frame, if there.
- “save-file” (*key events*) — Saves the buffer. If the buffer has no name, then `finder:put-file` is invoked.
- “save-file-as” (*key events*) — Calls `finder:put-file` to save the buffer.
- “load-file” (*key events*) — Invokes `finder:open-file`.
- “find-string” (*key events*) — Opens the search buffer at the bottom of the frame, unless it is already open, in which case it searches for the text in the search buffer.
- “find-string-reverse” (*key events*) — Same a “find-string”, but in the reverse direction.
- “find-string-replace” (*key events*) — Opens a replace string dialog box.
- “toggle-anchor” (*key events*) — Turns selection-anchoring on or off.
- “center-view-on-line” (*key events*) — Centers the buffer in its display using the currently selected line.
- “collapse-space” (*key events*) — Collapses all non-return whitespace around the caret into a single space.
- “remove-space” (*key events*) — Removes all non-return whitespace around the caret.
- “collapse-newline” (*key events*) — Collapses all empty lines around the caret into a single empty line. If there is only one empty line, it is removed.
- “open-line” (*key events*) — Inserts a new line.
- “transpose-chars” (*key events*) — Transposes the characters before and after the caret and moves forward one position.
- “transpose-words” (*key events*) — Transposes words before and after the caret and moves forward one word.
- “capitalize-word” (*key events*) — Changes the first character of the next word to a capital letter and moves to the end of the word.
- “upcase-word” (*key events*) — Changes all characters of the next word to capital letters and moves to the end of the word.
- “downcase-word” (*key events*) — Changes all characters of the next word to lowercase letters and moves to the end of the word.
- “kill-word” (*key events*) — Kills the next word.
- “backward-kill-word” (*key events*) — Kills the previous word.
- “goto-line” (*any events*) — Queries the user for a line number and moves the caret there.
- “goto-position” (*any events*) — Queries the user for a position number and moves the caret there.
- “copy-clipboard” (*mouse events*) — Copies the current selection to the clipboard.
- “cut-clipboard” (*mouse events*) — Cuts the current selection to the clipboard.
- “paste-clipboard” (*mouse events*) — Pastes the clipboard to the current selection.
- “copy-click-region” (*mouse events*) — Copies the region between the caret and the input mouse event.

- “cut-click-region” (*mouse events*) — Cuts the region between the caret and the input mouse event.
- “paste-click-region” (*mouse events*) — Pastes the clipboard into the position of the input mouse event.
- “select-click-word” (*mouse events*) — Selects the word under the input mouse event.
- “select-click-line” (*mouse events*) — Selects the line under the input mouse event.
- “start-macro” (*key events*) — Starts building a keyboard macro
- “end-macro” (*key events*) — Stops building a keyboard macro
- “do-macro” (*key events*) — Executes the last keyboard macro
- “toggle-overwrite” (*key events*) — Toggles overwriting mode

These functions are bound to the following keys (C = control, S = shift, A = alt, M = “meta”, D = command):

- C-g : “ring-bell”
- M-C-g : “ring-bell”
- C-c C-g : “ring-bell”
- C-x C-g : “ring-bell”
- C-p : “previous-line”
- S-C-p : “select-previous-line”
- C-n : “next-line”
- S-C-n : “select-next-line”
- C-e : “end-of-line”
- S-C-e : “select-to-end-of-line”
- D-RIGHT : “end-of-line”
- S-D-RIGHT : “select-to-end-of-line”
- M-RIGHT : “end-of-line”
- S-M-RIGHT : “select-to-end-of-line”
- C-a : “beginning-of-line”
- S-C-a : “select-to-beginning-of-line”
- D-LEFT : “beginning-of-line”
- D-S-LEFT : “select-to-beginning-of-line”
- M-LEFT : “beginning-of-line”
- M-S-LEFT : “select-to-beginning-of-line”
- C-h : “delete-previous-character”
- C-d : “delete-next-character”
- C-f : “forward-character”
- S-C-f : “select-forward-character”
- C-b : “backward-character”
- S-C-b : “select-backward-character”
- M-f : “forward-word”
- S-M-f : “select-forward-word”
- A-RIGHT : “forward-word”
- A-S-RIGHT : “forward-select-word”
- M-b : “backward-word”
- S-M-b : “select-backward-word”
- A-LEFT : “backward-word”
- A-S-LEFT : “backward-select-word”
- M-d : “kill-word”
- M-DELETE : “backward-kill-word”
- M-c : “capitalize-word”
- M-u : “upcase-word”
- M-l : “downcase-word”
- M-< : “beginning-of-file”
- S-M-< : “select-to-beginning-of-file”
- M-> : “end-of-file”
- S-M-> : “select-to-end-of-file”
- C-v : “next-page”
- S-C-v : “select-next-page”
- M-v : “previous-page”
- S-M-v : “select-previous-page”
- C-l : “center-view-on-line”
- C-k : “delete-to-end-of-line”
- C-y : “paste-clipboard” (Except Windows)
- A-v : “paste-clipboard”
- D-v : “paste-clipboard”
- C- : “undo”
- C-x u : “undo”
- C+ : “redo”
- C-w : “cut-clipboard”
- M-w : “copy-clipboard”
- C-x C-s : “save-file”
- C-x C-w : “save-file-as”
- C-x C-f : “load-file”
- C-s : “find-string”
- C-r : “find-string-reverse”
- M-% : “find-string-replace”
- SPACE : “collapse-space”
- M-\ : “remove-space”
- C-x C-o : “collapse-newline”
- C-o : “open-line”
- C-t : “transpose-chars”
- M-t : “transpose-words”

- C-SPACE : “toggle-anchor”
- M-g : “goto-line”
- M-p : “goto-position”
- LEFTBUTTONTRIPLE : “select-click-line”
- LEFTBUTTONDOUBLE : “select-click-word”
- RIGHTBUTTON : “copy-click-region”
- RIGHTBUTTONDOUBLE : “cut-click-region”
- MIDDLEBUTTON : “paste-click-region”
- C-RIGHTBUTTON : “copy-clipboard”
- INSERT : “toggle-overwrite”
- M-o : “toggle-overwrite”

keymap:setup-search

- (keymap:setup-search *keymap*) ⇒ void?
keymap : (is-a?/c **keymap%**)

This extends a **keymap%** with the bindings for searching.

number-snip:make-fraction-snip

- (number-snip:make-fraction-snip *num show-prefix-in-decimal-view?*) ⇒ (is-a?/c **snip%**)
num : number?
show-prefix-in-decimal-view? : boolean?

Makes a number snip that shows a fractional view of *number*. The boolean indicates if a #e prefix appears on the number, when shown in the decimal state.

See also **number-snip:make-repeating-decimal-snip**.

number-snip:make-repeating-decimal-snip

- (number-snip:make-repeating-decimal-snip *num show-prefix?*) ⇒ (is-a?/c **snip%**)
num : number?
show-prefix? : boolean?

Makes a number snip that shows the decimal expansion for *number*. The boolean indicates if a #e prefix appears on the number.

See also **number-snip:make-fraction-snip**.

path-utils:generate-autosave-name

- (path-utils:generate-autosave-name *filename*) ⇒ string?
filename : string?

Generates a name for an autosave file from *filename*.

path-utils:generate-backup-name

- (path-utils:generate-backup-name *filename*) ⇒ path?
filename: path?

Generates a name for an backup file from *filename*.

preferences:add-callback

- (preferences:add-callback *p f weak?*) ⇒ (-> void?)
p: symbol?
f: (symbol? any/c .-> . any/c)
weak? = #f: boolean?

This function adds a callback which is called with a symbol naming a preference and it's value, when the preference changes. `preferences:add-callback` returns a thunk, which when invoked, removes the callback from this preference.

If *weak?* is true, the preferences system will only hold on to the callback weakly.

The callbacks will be called in the order in which they were added.

If you are adding a callback for a preference that requires marshalling and unmarshalling, you must set the marshalling and unmarshalling functions by calling `preferences:set-un/marshall` before adding a callback.

This function raises `exn:unknown-preference` if the preference has not been set.

preferences:add-can-close-dialog-callback

- (preferences:add-can-close-dialog-callback *cb*) ⇒ void?
cb: (-> boolean?)

Registers *cb*. Next time the user clicks the OK button the preferences dialog, all of the *cb* functions are called. If any of them return `\#f`, the dialog is not closed.

See also `preferences:add-on-close-dialog-callback`.

preferences:add-editor-checkbox-panel

- (preferences:add-editor-checkbox-panel) ⇒ void?
 Adds a preferences panel for configuring options related to editing.

preferences:add-font-panel

- (preferences:add-font-panel) ⇒ void?

Adds a font selection preferences panel to the preferences dialog.

preferences:add-on-close-dialog-callback

- (preferences:add-on-close-dialog-callback *cb*) ⇒ void?
cb : (-> void?)

Registers *cb*. Next time the user clicks the OK button the preferences dialog, all of the *cb* functions are called, assuming that each of the callbacks passed to `preferences:add-can-close-dialog-callback` succeed.

preferences:add-panel

- (preferences:add-panel *labels* *f*) ⇒ void?
labels : (union string? (cons/c string? (listof string?)))
f : ((is-a?/c `area-container-window`<%>) . ->d . ((parent) (let ((children (map ((x) (x) (send parent get-children)))) ((child) (and (is-a? child `area-container-window`<%>) (andmap eq? (append children (list child)) (send parent get-children)))))))

preferences:add-preference-panel adds the result of *f* with name *labels* to the preferences dialog box.

The labels determine where this preference panel is placed in the dialog. If the list is just one string, the preferences panel is placed at the top level of the dialog. If there are more strings, a hierarchy of nested panels is created and the new panel is added at the end. If multiple calls to `preferences:add-preference-panel` pass the same prefix of strings, those panels are placed in the same children.

When the preference dialog is opened for the first time, the function *f* is called with a panel, and *f* is expected to add a new child panel to it and add whatever preferences configuration controls it wants to that panel. Then, *f*'s should return the panel it added.

preferences:add-scheme-checkbox-panel

- (preferences:add-scheme-checkbox-panel) ⇒ void?

Adds a preferences panel for configuring options related to Scheme.

preferences:add-to-editor-checkbox-panel

- (preferences:add-to-editor-checkbox-panel *proc*) ⇒ void?
proc : ((is-a?/c `vertical-panel`%) . -> . void?)

Saves *proc* until the preferences panel is created, when it is called with the Echeme preferences panel to add new children to the panel.

`preferences:add-to-scheme-checkbox-panel`

- `(preferences:add-to-scheme-checkbox-panel proc) ⇒ void?`
`proc: ((is-a?/c vertical-panel%) .-> . void?)`

Saves *proc* until the preferences panel is created, when it is called with the Scheme preferences panel to add new children to the panel.

`preferences:add-to-warnings-checkbox-panel`

- `(preferences:add-to-warnings-checkbox-panel proc) ⇒ void?`
`proc: ((is-a?/c vertical-panel%) .-> . void?)`

Saves *proc* until the preferences panel is created, when it is called with the Misc. panel to add new children to the panel.

`preferences:add-warnings-checkbox-panel`

- `(preferences:add-warnings-checkbox-panel) ⇒ void?`
 Adds a preferences panel for configuring options relating to warnings

`preferences:get`

- `(preferences:get symbol) ⇒ any/c`
`symbol: symbol?`

See also `preferences:set-default`.

`preferences:get` returns the value for the preference *symbol*. It raises `exn:unknown-preference` if the preference's default has not been set.

`preferences:hide-dialog`

- `(preferences:hide-dialog) ⇒ void?`
 Hides the preferences dialog.

`preferences:restore-defaults`

- `(preferences:restore-defaults) ⇒ void?`
`(preferences:restore-defaults)` restores the users's configuration to the default preferences.

`preferences:save`

- (`preferences:save`) ⇒ `boolean?`
`(preferences:save-user-preferences)` saves the user's preferences to disk, potentially marshalling some of the preferences.
 Returns `#f` if saving the preferences fails and `#t` otherwise.

`preferences:set`

- (`preferences:set symbol value`) ⇒ `void?`
`symbol`: `symbol?`
`value`: `any/c`
 See also `preferences:set-default`.
`preferences:set-preference` sets the preference `symbol` to `value`. This should be called when the users requests a change to a preference.
 It raises `exn:unknown-preference` if the preference's default has not been set.

`preferences:set-default`

- (`preferences:set-default symbol value test`) ⇒ `void?`
`symbol`: `symbol?`
`value`: `any/c`
`test`: `(any/c .-> . any)`
 This function must be called every time your application starts up, before any call to `preferences:get`, `preferences:set` (for any given preference).
 If you use `preferences:set-un/marshall`, you must call this function before calling it.
 This sets the default value of the preference `symbol` to `value`. If the user has chosen a different setting, the user's setting will take precedence over the default value.
 The last argument, `test` is used as a safeguard. That function is called to determine if a preference read in from a file is a valid preference. If `test` returns `#t`, then the preference is treated as valid. If `test` returns `#f` then the default is used.

`preferences:set-un/marshall`

- (`preferences:set-un/marshall symbol marshall unmarshall`) ⇒ `void?`
`symbol`: `symbol?`
`marshall`: `(any/c .-> . printable/c)`
`unmarshall`: `(printable/c .-> . any/c)`
`preferences:set-un/marshall` is used to specify marshalling and unmarshalling functions for the preference `symbol`. `marshall` will be called when the users saves their preferences to turn the preference value

for *symbol* into a printable value. *unmarshall* will be called when the user's preferences are read from the file to transform the printable value into its internal representation. If `preference:set-un/marshall` is never called for a particular preference, the values of that preference are assumed to be printable.

If the unmarshalling function returns a value that does not meet the guard passed to `preferences:set-default` for this preference, the default value is used.

The *marshall* function might be called with any value returned from `read` and it must not raise an error (although it can return arbitrary results if it gets bad input). This might happen when the preferences file becomes corrupted, or is edited by hand.

`preference:set-un/marshall` must be called before calling `preferences:get`, `preferences:set`.

`preferences:show-dialog`

- (`preferences:show-dialog`) ⇒ `void?`

Shows the preferences dialog.

`preferences:silent-save`

- (`preferences:silent-save`) ⇒ `boolean?`

Same as `preferences:save` except that it does not put display a message if it fails.

`scheme:add-coloring-preferences-panel`

- (`scheme:add-coloring-preferences-panel`) ⇒ `any`

Installs the "Scheme" preferences panel in the "Syntax Coloring" section.

`scheme:add-preferences-panel`

- (`scheme:add-preferences-panel`) ⇒ `void?`

Adds a tabbing preferences panel to the preferences dialog.

`scheme:get-color-prefs-table`

- (`scheme:get-color-prefs-table`) ⇒ (`listof (list/c symbol? (is-a?/c color%))`)

Returns a table mapping from symbols (naming the categories that the online colorer uses for Scheme mode coloring) to their colors.

These symbols are suitable for input to `scheme:short-sym->pref-name` and `scheme:short-sym->style-name`.

`scheme:get-keymap`

- `(scheme:get-keymap) ⇒ (is-a?/c keymap%)`
Returns a keymap with binding suitable for Scheme.

`scheme:get-wordbreak-map`

- `(scheme:get-wordbreak-map) ⇒ (is-a?/c editor-wordbreak-map%)`
This method returns a `editor-wordbreak-map%` that is suitable for Scheme.

`scheme:init-wordbreak-map`

- `(scheme:init-wordbreak-map key) ⇒ void?`
`key: (is-a?/c keymap%)`
Initializes the workdbreak map for `keymap`.

`scheme:setup-keymap`

- `(scheme:setup-keymap keymap) ⇒ void?`
`keymap: (is-a?/c keymap%)`
Initializes `keymap` with Scheme-mode keybindings.

`scheme:short-sym->pref-name`

- `(scheme:short-sym->pref-name short-sym) ⇒ symbol?`
`short-sym: symbol?`
Builds the symbol naming the preference from one of the symbols in the table returned by `scheme:get-color-prefs-table`.

`scheme:short-sym->style-name`

- `(scheme:short-sym->style-name short-sym) ⇒ string?`
`short-sym: symbol?`
Builds the symbol naming the editor style from one of the symbols in the table returned by `scheme:get-color-prefs-table`. This style is a named style in the style list returned by `editor:get-standard-style-list`.

`scheme:text-balanced?`

- (`scheme:text-balanced?` *text start end*) ⇒ `boolean?`
`text`: (`is-a?/c text%`)
`start = 0`: `number?`
`end = #f`: (`union false/c number?`)

Determines if the range in the editor from *start* to *end* in *text* is a matched set of parenthesis. If *end* is `#f`, it defaults to the last position of the *text*.

The implementation of this function creates a port with `open-input-text-editor` and then uses ‘read’ to parse the range of the buffer.

`test:button-push`

- (`test:button-push` *button*) ⇒ `void?`
`button`: (`union` (`str`) (`and` (`string?` `str`) (`test:top-level-focus-window-has?` (`c`) (`and` (`is-a?` `c button%`) (`string=?` (`send` `c get-label`) `str`) (`send` `c is-enabled?`) (`send` `c is-shown?`)))))) (`and/c` (`is-a?` `c button%`) (`btn`) (`and` (`send` `btn is-enabled?`) (`send` `btn is-shown?`))) (`btn`) (`test:top-level-focus-window-has?` (`c`) (`eq?` `c btn`))))))

Simulates pushing *button*. If a string is supplied, the primitive searches for a button labelled with that string in the active frame. Otherwise, it pushes the button argument.

`test:close-top-level-window`

- (`test:close-top-level-window` *tlw*) ⇒ `void?`
`tlw`: (`is-a?/c top-level-window<%>`)

Use this function to simulate clicking on the close box of a frame. Closes *tlw* with this expression:

```
(when (send tlw can-close?)
      (send tlw on-close)
      (send tlw show #f))
```

`test:current-get-eventspaces`

- (`test:current-get-eventspaces` *func*) ⇒ `void?`
`func`: (`->` (`listof eventspace?`))
- (`test:current-get-eventspaces`) ⇒ (`->` (`listof eventspace?`))

This parameter that specifies which `eventspaceinfo` are considered when finding the frontmost frame. The first case sets the parameter to *func*. The procedure *func* will be invoked with no arguments to determine the eventspaces to consider when finding the frontmost frame for simulated user events. The second case returns the current value of the parameter. This will be a procedure which, when invoked, returns a list of eventspaces.

test:keystroke

- (test:keystroke *key* *modifier-list*) ⇒ void?
 - key*: (union char? symbol?)
 - modifier-list* = null: (listof (symbols (quote alt) (quote control) (quote meta) (quote shift) (quote noalt) (quote nocontrol) (quote nometa) (quote noshift)))

This function simulates a user pressing a key. The argument, *key*, is just like the argument to the `get-key-code` method of the `key-event%` class.

Note: To send the “Enter” key, use `# eturn`, not `# ewline`.

The `'shift` or `'noshift` modifier is implicitly set from *key*, but is overridden by the argument list. The `'shift` modifier is set for any capitol alpha-numeric letters and any of the following characters:

```
#\? #\: #\~ #\\ #\|
#\< #\> #\{ #\} #\[ #\] #\(\ #\)
#\! #\@ #\# #\$ #\% #\^ #\& #\* #\_ #\+
```

If conflicting modifiers are provided, the ones later in the list are used.

test:menu-select

- (test:menu-select *menu* *item*) ⇒ void?
 - menu*: string?
 - item*: string?

Selects the menu-item named *item* in the menu named *menu*.

Note: The string for the menu item does not include its keyboard equivalent. For example, to select “New” from the “File” menu, use “New”, not “New Ctrl+m n”.

test:mouse-click

- (test:mouse-click *button* *x* *y* *modifiers*) ⇒ void?
 - button*: (symbols 'left 'middle 'right)
 - x*: (and/c exact? integer?)
 - y*: (and/c exact? integer?)
 - modifiers* = null: (listof (symbols (quote alt) (quote control) (quote meta) (quote shift) (quote noalt) (quote nocontrol) (quote nometa) (quote noshift)))

Simulates a mouse click at the coordinate: (*x*,*y*) in the currently focused `window<%>`, assuming that it supports the `on-event` method. Use `test:button-push` to click on a button.

On the Macintosh, `'right` corresponds to holding down the command modifier key while clicking and `'middle` cannot be generated.

Under Windows, `'middle` can only be generated if the user has a three button mouse.

The modifiers later in the list *modifiers* take precedence over ones that appear earlier.

test:new-window

- (test:new-window *window*) ⇒ void?
window: (is-a?/c window<*>)

Moves the keyboard focus to a new window within the currently active frame. Unfortunately, neither this function nor any other function in the test engine can cause the focus to move from the top-most (active) frame.

test:number-pending-actions

- (test:number-pending-actions) ⇒ number?
Returns the number of pending events (those that haven't completed yet)

test:reraise-error

- (test:reraise-error) ⇒ void?
See also [fw:test:errors](#).

test:run-interval

- (test:run-interval *msec*) ⇒ void?
msec: number?
- (test:run-interval) ⇒ number?

See also [fw:actions-completeness](#). The first case in the case-lambda sets the run interval to *msec* milliseconds and the second returns the current setting.

test:run-one

- (test:run-one *f*) ⇒ void?
f: (-> void?)

Runs the function *f* as if it was a simulated event. See also [fw:test](#).

test:set-check-box!

- (test:set-check-box! *check-box state*) ⇒ void?
check-box : (union string? (is-a?/c **check-box%**))
state : boolean?

Clears the **check-box%** item if *state* is #f, and sets it otherwise.

If *check-box* is a string, this function searches for a **check-box%** with a label matching that string, otherwise it uses *check-box* itself.

test:set-choice!

- (test:set-choice! *choice str*) ⇒ void?
choice : (union string? (is-a?/c **choice%**))
str : string?

Selects *choice*'s item *str*. If *choice* is a string, this function searches for a **choice%** with a label matching that string, otherwise it uses *choice* itself.

test:set-radio-box!

- (test:set-radio-box! *radio-box state*) ⇒ void?
radio-box : (union string? (is-a?/c **radio-box%**))
state : (union string? number?)

Sets the radio-box to *state*. If *state* is a string, this function finds the choice with that label and if it is a number, it uses the number as an index into the state. If the number is out of range or if the label isn't in the radio box, an exception is raised.

If *radio-box* is a string, this function searches for a **radio-box%** with a label matching that string, otherwise it uses *radio-box* itself.

test:set-radio-box-item!

- (test:set-radio-box-item! *entry*) ⇒ void?
entry : string?

Finds a **radio-box%** that has a label *entry* and sets the radio-box to *entry*.

test:top-level-focus-window-has?

- (test:top-level-focus-window-has? *test*) ⇒ boolean?
test : ((is-a?/c **area<%>**) .-> . boolean?)

Calls *test* for each child of the top-level-focus-frame and returns #t if *test* ever does, otherwise returns #f. If there is no top-level-focus-window, returns #f.

`version:add-spec`

- `(version:add-spec spec revision) ⇒ void?`
 `spec: any/c`
 `revision: any/c`

These two values are appended to the version string. `write` is used to transform them to strings. For example:

```
(version:add-spec 's 1)
```

in version 205 will make the version string be "205s1". The symbols `'f` and `'d` are used internally for framework and drscheme revisions.

`version:version`

- `(version:version) ⇒ string?`

This function returns a string describing the version of this application. See also `version:add-spec`.

Index

- 'framework:backup-files?', 35
- 'framework:basic-canvas-background', 10

- active-child, 167
- add-tall-snip, 12, 201
- add-wide-snip, 12, 201
- after-change-style, 22, 133, 142, 156, 208
- after-delete, 22, 134, 143, 156, 208, 212
- after-edit-sequence, 22, 29, 134, 143, 157, 209
- after-insert, 22, 134, 144, 157, 195, 199, 209, 212
- after-io-insertion, 217
- after-load-file, 29, 134, 144, 157, 209, 215
- after-new-child, 48, 167, 171
- after-percentage-change, 170
- after-save-file, 29, 134, 144, 157, 215
- after-set-position, 22, 135, 145, 157, 198, 213
- after-set-size-constraint, 135, 145, 158
- alignment, 46, 112–118, 169, 174
- allow-close-with-no-filename?, 33
- anchor-status-changed, 57
- application:current-app-name, 230
- 'auto-hscroll, 9, 14–16
- 'auto-vscroll, 9, 14–16
- auto-wrap, 23, 142, 189, 195, 198, 214, 224–228
- autosave:autosavable<%>, 7
- autosave:register, 230
- autosave:restore-autosave-files/gui, 230
- autosave?, 35
- autosaving, 7

- background color, 10
- backup?, 35
- backward-containing-sexp, 17
- “backward-kill-word”, 252
- backward-match, 17
- backward-sexp, 182
- balance-parens, 182
- 'border, 169, 174
- border, 46, 112–118, 169, 174
- box-comment-out-selection, 182

- callback, 131, 132
- can-change-style?, 135, 145, 158
- can-close-all?, 120
- can-close?, 26, 34, 50, 95
- can-delete?, 135, 146, 158, 216, 222
- can-do-edit-operation?, 135, 146, 158
- can-exit?, 48
- can-insert?, 136, 147, 159, 216, 222
- can-load-file?, 136, 147, 159
- can-replace?, 106
- can-save-file?, 30, 136, 147, 159
- can-set-size-constraint?, 137, 148, 159
- canvas
 - scroll bars, 9
- canvas:basic-mixin, 9
- canvas:basic<%>, 9
- canvas:basic%, 13
- canvas:color-mixin, 10
- canvas:color<%>, 10
- canvas:color%, 14
- canvas:delegate-mixin, 11
- canvas:delegate<%>, 10
- canvas:delegate%, 15
- canvas:info-mixin, 11
- canvas:info<%>, 11
- canvas:info%, 14
- canvas:wide-snip-mixin, 13
- canvas:wide-snip<%>, 12
- canvas:wide-snip%, 16
- “capitalize-word”, 252
- “center-view-on-line”, 252
- chain-to-keymap, 127
- checked, 132
- classify-position, 18
- clear, 120
- clear-box-input-port, 217
- clear-input-port, 217
- clear-output-ports, 217
- close, 26, 44
- close-status-line, 52
- “collapse-newline”, 252
- “collapse-space”, 252
- color-model:rgb->xyz, 231
- color-model:rgb-color-distance, 231
- color-model:xyz->rgb, 231
- color-model:xyz-x, 231
- color-model:xyz-y, 232
- color-model:xyz-z, 232
- color-model:xyz?, 232
- color-prefs:add-background-preferences-panel, 232
- color-prefs:add-to-preferences-panel, 232

- color-prefs:build-color-selection-panel, 232
- color-prefs:marshall-style, 233
- color-prefs:register-color-pref, 233
- color-prefs:unmarshall-style, 233
- color:text-mixin, 21
- color:text-mode-mixin, 24
- color:text-mode<%>, 24
- color:text-mode%, 24
- color:text<%>, 17
- color:text%, 23
- 'combo, 9, 14–16
- comment-out-selection, 183
- container-size, 168, 169, 172
- 'control-border, 9, 14–16
- 'copy, 37, 214
- copy, 180, 203, 206
- “copy-click-region”, 252
- “copy-clipboard”, 252
- “cut-click-region”, 253
- “cut-clipboard”, 252

- delegate-moved, 103
- delegated-text-shown?, 103
- delete/io, 217
- 'deleted, 9, 14–16, 169, 174
- demand-callback, 131, 132
- determine-width, 53
- do-autosave, 7, 35
- “do-macro”, 253
- down-sexp, 183
- “downcase-word”, 252
- drag-and-drop, 48
- draw, 180, 203, 206

- edit-menu:after-preferences, 60
- edit-menu:between-clear-and-select-all, 60
- edit-menu:between-copy-and-paste, 61
- edit-menu:between-cut-and-copy, 61
- edit-menu:between-find-and-preferences, 61
- edit-menu:between-paste-and-clear, 61
- edit-menu:between-redo-and-cut, 61
- edit-menu:between-select-all-and-find, 62, 95
- edit-menu:clear-callback, 62
- edit-menu:clear-help-string, 62
- edit-menu:clear-on-demand, 62
- edit-menu:clear-string, 63
- edit-menu:copy-callback, 63
- edit-menu:copy-help-string, 63
- edit-menu:copy-on-demand, 63
- edit-menu:copy-string, 64
- edit-menu:create-clear?, 64
- edit-menu:create-copy?, 64
- edit-menu:create-cut?, 64
- edit-menu:create-find-again?, 64, 108
- edit-menu:create-find?, 65, 108
- edit-menu:create-paste?, 65
- edit-menu:create-preferences?, 65
- edit-menu:create-redo?, 65
- edit-menu:create-replace-and-find-again?, 65, 109
- edit-menu:create-select-all?, 66
- edit-menu:create-undo?, 66
- edit-menu:cut-callback, 66
- edit-menu:cut-help-string, 66
- edit-menu:cut-on-demand, 67
- edit-menu:cut-string, 67
- edit-menu:find-again-callback, 67, 109
- edit-menu:find-again-help-string, 67
- edit-menu:find-again-on-demand, 67
- edit-menu:find-again-string, 68
- edit-menu:find-callback, 68, 109
- edit-menu:find-help-string, 68
- edit-menu:find-on-demand, 68
- edit-menu:find-string, 69
- edit-menu:get-clear-item, 69
- edit-menu:get-copy-item, 69
- edit-menu:get-cut-item, 69
- edit-menu:get-find-again-item, 69
- edit-menu:get-find-item, 69
- edit-menu:get-paste-item, 70
- edit-menu:get-preferences-item, 70
- edit-menu:get-redo-item, 70
- edit-menu:get-replace-and-find-again-item, 70
- edit-menu:get-select-all-item, 70
- edit-menu:get-undo-item, 70
- edit-menu:paste-callback, 71
- edit-menu:paste-help-string, 71
- edit-menu:paste-on-demand, 71
- edit-menu:paste-string, 71
- edit-menu:preferences-callback, 72
- edit-menu:preferences-help-string, 72
- edit-menu:preferences-on-demand, 72
- edit-menu:preferences-string, 72
- edit-menu:redo-callback, 72
- edit-menu:redo-help-string, 73
- edit-menu:redo-on-demand, 73
- edit-menu:redo-string, 73
- edit-menu:replace-and-find-again-callback, 73, 109
- edit-menu:replace-and-find-again-help-string, 74
- edit-menu:replace-and-find-again-on-demand,

- 74, 109
- edit-menu:replace-and-find-again-string, 74
- edit-menu:select-all-callback, 74
- edit-menu:select-all-help-string, 75
- edit-menu:select-all-on-demand, 75
- edit-menu:select-all-string, 75
- edit-menu:undo-callback, 75
- edit-menu:undo-help-string, 76
- edit-menu:undo-on-demand, 76
- edit-menu:undo-string, 76
- editing-this-file?, 44, 95
- editor, 9, 13–16
- editor-position-changed, 57
- editor:autowrap-mixin, 33
- editor:autowrap<%>, 33
- editor:backup-autosave-mixin, 36
- editor:backup-autosave<%>, 35
- editor:basic-mixin, 28
- editor:basic<%>, 26
- editor:file-mixin, 34
- editor:file<%>, 33
- editor:get-default-color-style-name, 233
- editor:get-standard-style-list, 234
- editor:info-mixin, 37
- editor:info<%>, 37
- editor:keymap-mixin, 32
- editor:keymap<%>, 32
- editor:set-default-font-color, 234
- editor:set-standard-style-list-delta, 234
- editor:set-standard-style-list-pref-callbacks, 234
- editor:standard-style-list-mixin, 31
- editor:standard-style-list<%>, 31
- editors
 - events, 149, 151–153, 201, 223, 224
 - hooks, 22, 23, 29, 30, 36, 142–148, 150, 152, 155, 195, 198, 199, 208–210, 212, 213, 216, 222
 - locking, 23, 38
 - modified, 37
- enabled, 9, 13–16, 46, 112–118, 169, 174
- “end-macro”, 253
- erase-underscores, 131
- exit callbacks, 39
- exit:can-exit?, 234
- exit:exit, 234
- exit:exiting?, 235
- exit:insert-can?-callback, 235
- exit:insert-on-callback, 235
- exit:on-exit, 235
- exit:set-exiting, 235
- exit:user-oks-exit, 236
- exiting, 39
- exn:exn?, 236
- exn:make-exn, 236
- exn:make-unknown-preference, 236
- exn:unknown-preference, 255, 257, 258
- exn:unknown-preference?, 236
- file-menu:after-quit, 76
- file-menu:between-close-and-quit, 76
- file-menu:between-new-and-open, 77
- file-menu:between-open-and-revert, 77
- file-menu:between-print-and-close, 77
- file-menu:between-revert-and-save, 77
- file-menu:between-save-as-and-print, 77
- file-menu:close-callback, 78
- file-menu:close-help-string, 78
- file-menu:close-on-demand, 78
- file-menu:close-string, 78
- file-menu:create-close?, 78
- file-menu:create-new?, 79
- file-menu:create-open-recent?, 79
- file-menu:create-open?, 79
- file-menu:create-print?, 79, 95
- file-menu:create-quit?, 80
- file-menu:create-revert?, 80, 95
- file-menu:create-save-as?, 80, 96
- file-menu:create-save?, 80, 96
- file-menu:get-close-item, 80
- file-menu:get-new-item, 81
- file-menu:get-open-item, 81
- file-menu:get-open-recent-item, 81
- file-menu:get-print-item, 81
- file-menu:get-quit-item, 81
- file-menu:get-revert-item, 81
- file-menu:get-save-as-item, 82
- file-menu:get-save-item, 82
- file-menu:new-callback, 82, 100
- file-menu:new-help-string, 82
- file-menu:new-on-demand, 82, 100
- file-menu:new-string, 83
- file-menu:open-callback, 83
- file-menu:open-help-string, 83
- file-menu:open-on-demand, 83, 100
- file-menu:open-recent-callback, 83
- file-menu:open-recent-help-string, 84
- file-menu:open-recent-on-demand, 84
- file-menu:open-recent-string, 84
- file-menu:open-string, 84
- file-menu:print-callback, 84, 96
- file-menu:print-help-string, 85
- file-menu:print-on-demand, 85
- file-menu:print-string, 85
- file-menu:quit-callback, 85
- file-menu:quit-help-string, 85

- file-menu:quit-on-demand, 86
- file-menu:quit-string, 86
- file-menu:revert-callback, 86, 96
- file-menu:revert-help-string, 86
- file-menu:revert-on-demand, 86, 96
- file-menu:revert-string, 87
- file-menu:save-as-callback, 87, 97
- file-menu:save-as-help-string, 87
- file-menu:save-as-on-demand, 87
- file-menu:save-as-string, 87
- file-menu:save-callback, 88, 97
- file-menu:save-help-string, 88
- file-menu:save-on-demand, 88
- file-menu:save-string, 88
- filename, 94, 116–118
- files
 - formats, 124
 - loading, 29, 144, 147, 152, 209, 210, 215
 - names, 35
 - opening, 124
 - saving, 29, 30, 37, 144, 147, 155, 214, 215
 - selecting, 41
- find-down-sexp, 183
- “find-string”, 252
- “find-string-replace”, 252
- “find-string-reverse”, 252
- find-up-sexp, 183
- finder:common-get-file, 236
- finder:common-get-file-list, 237
- finder:common-put-file, 237
- finder:default-extension, 237
- finder:default-filters, 238
- finder:get-file, 238
- finder:open-file, 252
- finder:put-file, 238, 252
- finder:std-get-file, 239
- finder:std-put-file, 239
- flash-backward-sexp, 183
- flash-forward-sexp, 184
- ‘float, 46, 112–119
- for-each-frame, 120
- force-stop-colorer, 18
- format handler, 124
- forward-match, 18
- forward-sexp, 184
- frame groups, 120
- frame-label-changed, 121
- frame-shown/hidden, 121
- frame:add-snip-menu-items, 239
- frame:basic-mixin, 46
- frame:basic<%, 44
- frame:basic%, 112
- frame:delegate-mixin, 104
- frame:delegate<%, 103
- frame:delegate%, 118
- frame:editor-mixin, 94
- frame:editor<%, 92
- frame:editor%, 116
- frame:info-mixin, 55
- frame:info<%, 53
- frame:info%, 113
- frame:open-here-mixin, 99
- frame:open-here<%, 99
- frame:open-here%, 116
- frame:pasteboard-info-mixin, 58
- frame:pasteboard-info<%, 58
- frame:pasteboard-info%, 114
- frame:pasteboard-mixin, 102
- frame:pasteboard<%, 102
- frame:pasteboard%, 118
- frame:register-group-mixin, 50
- frame:register-group<%, 50
- frame:reorder-menus, 239
- frame:searchable-mixin, 108
- frame:searchable-text-mixin, 111
- frame:searchable-text<%, 111
- frame:searchable<%, 105
- frame:searchable%, 117
- frame:setup-size-pref, 240
- frame:size-pref-mixin, 49
- frame:size-pref<%, 49
- frame:size-pref%, 112
- frame:standard-menus-mixin, 91
- frame:standard-menus<%, 58
- frame:standard-menus%, 115
- frame:status-line-mixin, 52
- frame:status-line<%, 51
- frame:status-line%, 114
- frame:text-info-mixin, 57
- frame:text-info<%, 56
- frame:text-info%, 113
- frame:text-mixin, 101
- frame:text<%, 101
- frame:text%, 117
- framework-class^, 3
- framework-sig.ss, 3
- framework-unit.ss, 3
- framework.ss, 3
- framework^, 3
- freeze-colorer, 18
- get-active-frame, 121
- get-allow-edits, 218
- get-area-container, 44
- get-area-container%, 45
- get-backward-sexp, 184
- get-box-input-editor-snip%, 218

- get-box-input-text%, 218
- get-can-close-parent, 33
- get-canvas, 92
- get-canvas<%>, 92
- get-canvas%, 92
- get-chained-keymaps, 126
- get-checkable-menu-item%, 88
- get-delegate, 202
- get-delegated-text, 103
- get-edit-menu, 89
- get-editor, 92
- get-editor<%>, 92, 101, 102, 104, 111
- get-editor%, 93, 102, 105, 111
- get-entire-label, 93
- get-err-port, 218
- get-err-style-delta, 218
- get-extent, 180, 204, 206
- get-file, 30
- get-file-menu, 89
- get-filename, 45, 97
- get-filename/untitled-name, 26
- get-fixed-style, 193, 197
- get-forward-sexp, 184
- get-frames, 121
- get-help-menu, 89
- get-highlighted-ranges, 193
- get-in-box-port, 218
- get-in-port, 219
- get-info-canvas, 54
- get-info-editor, 54
- get-info-panel, 54
- get-insertion-point, 219
- get-keymaps, 32, 34, 200
- get-label, 97
- get-label-prefix, 93
- get-limit, 184
- get-map-function-table, 126
- get-map-function-table/ht, 126
- get-mdi-parent, 121
- get-menu-bar%, 45
- get-menu-item%, 89
- get-menu%, 89
- get-open-here-editor, 99
- get-open-here-frame, 122
- get-out-port, 219
- get-out-style-delta, 219
- get-percentages, 170
- get-read-write?, 215
- get-saved-snips, 179
- get-styles-fixed, 193
- get-surrogate, 133
- get-tab-size, 184
- get-text, 181
- get-text-to-search, 106, 112
- get-top-level-window, 27
- get-unread-start-point, 219
- get-value-port, 219
- get-value-style-delta, 220
- get-vertical?, 170, 173
- “goto-line”, 252
- “goto-position”, 252
- group:%, 120
- group:get-the-frame-group, 240
- 'guess, 37, 214
- gui-utils.ss, 3
- gui-utils:cancel-on-right?, 240
- gui-utils:cursor-delay, 240
- gui-utils:delay-action, 241
- gui-utils:get-choice, 241
- gui-utils:get-clickback-delta, 242
- gui-utils:get-clicked-clickback-delta, 242
- gui-utils:local-busy-cursor, 242
- gui-utils:next-untitled-name, 242
- gui-utils:ok/cancel-buttons, 242
- gui-utils:show-busy-cursor, 243
- gui-utils:trim-string, 243
- gui-utils:unsaved-warning, 243
- handler:add-to-recent, 243
- handler:current-create-new-window, 244
- handler:edit-file, 244
- handler:find-format-handler, 244
- handler:find-named-format-handler, 245
- handler:handler-extension, 245
- handler:handler-handler, 245
- handler:handler-name, 245
- handler:handler?, 245
- handler:insert-format-handler, 245
- handler:install-recent-items, 246
- handler:open-file, 246
- handler:set-recent-items-frame-superclass, 246
- handler:set-recent-position, 246
- handler:size-recently-opened-files, 247
- has-focus?, 27
- height, 46, 112–118
- help-menu:about-callback, 90, 98
- help-menu:about-help-string, 90
- help-menu:about-on-demand, 90
- help-menu:about-string, 90, 98
- help-menu:after-about, 90
- help-menu:before-about, 91
- help-menu:create-about?, 91, 98
- help-menu:get-about-item, 91
- help-string, 131, 132
- hide-delegated-text, 103

- 'hide-hscroll, 9, 14–16
- hide-info, 54
- 'hide-menu-bar, 46, 112–119
- hide-search, 106
- 'hide-vscroll, 9, 14–16
- highlight-range, 193, 209
- horiz-margin, 9, 13–16, 169, 174
- horizontal-inset, 9, 13–16

- icon:get-anchor-bitmap, 247
- icon:get-autowrap-bitmap, 247
- icon:get-eof-bitmap, 247
- icon:get-gc-off-bitmap, 247
- icon:get-gc-on-bitmap, 247
- icon:get-left/right-cursor, 248
- icon:get-lock-bitmap, 248
- icon:get-paren-highlight-bitmap, 248
- icon:get-unlock-bitmap, 248
- icon:get-up/down-cursor, 248
- initial-autowrap-bitmap, 194
- insert, 204
- insert-before, 220
- insert-between, 220
- insert-close-paren, 19
- insert-frame, 122
- insert-return, 185
- is-frozen?, 19
- is-info-hidden?, 54
- is-stopped?, 19

- keyboard focus
 - notification, 12, 23, 31, 151
- keyboard mapping, 126
- keymap:add-to-right-button-menu, 248
- keymap:add-to-right-button-menu/before, 249
- keymap:add-user-keybindings-file, 249
- keymap:aug-keymap-mixin, 127
- keymap:aug-keymap<%>, 126
- keymap:aug-keymap%, 128
- keymap:call/text-keymap-initializer, 249
- keymap:canonicalize-keybinding-string, 249
- keymap:get-editor, 250
- keymap:get-file, 250
- keymap:get-global, 250
- keymap:get-search, 250
- keymap:make-meta-prefix-list, 250
- keymap:remove-chained-keymap, 251
- keymap:remove-user-keybindings-file, 251
- keymap:send-map-function-meta, 251
- keymap:set-chained-keymaps, 251
- keymap:setup-editor, 252
- keymap:setup-file, 252

- keymap:setup-global, 252
- keymap:setup-search, 254
- keymaps, 126
 - chaining, 127
 - in an editor, 142, 195, 198, 214, 224
- “kill-word”, 252

- label, 9, 13–16, 46, 112–115, 131, 132
- line-count, 9, 13–16
- line-spacing, 23, 142, 189, 195, 198, 214, 224–228
- “load-file”, 252
- load-file/gui-error, 27
- local-edit-sequence?, 27
- locate-file, 122
- lock, 23, 38
- lock-status-changed, 55

- make-editor, 93
- make-root-area-container, 45, 53, 56, 105, 110
- make-visible, 46
- map-function, 127
- mark-matching-parenthesis, 185
- 'mdi-child, 46, 112–119
- 'mdi-parent, 46, 112–119
- menu:can-restore-checkable-menu-item%, 132
- menu:can-restore-menu-item%, 131
- menu:can-restore-mixin, 130
- menu:can-restore-underscore-menu%, 132
- menu:can-restore-underscore-mixin, 131
- menu:can-restore-underscore<%>, 131
- menu:can-restore<%>, 130
- menus
 - standard editor items, 146
- Meta, 251
 - 'metal, 46, 112–119
- min-height, 9, 13–16, 46, 112–118, 169, 174
- min-width, 9, 13–16, 46, 112–118, 169, 174
- (mixin (dom;%_{*i*} ...) (rng;%_{*i*} ...) class-body-expr ...), 3
- mode:host-text-mixin, 142
- mode:host-text<%>, 133
- mode:surrogate-text<%>, 133
- mode:surrogate-text%, 156
- mouse mapping, 126
- move-to-search-or-reverse-search, 106
- move-to-search-or-search, 106
- move/copy-to-edit, 194
- '|MrEd:wheelStep|, 10

- 'no-border, 9, 14–16
- 'no-caption, 46, 112–119

- 'no-caret, 196, 211
- 'no-hscroll, 9, 14–16
- 'no-resize-border, 46, 112–119
- 'no-system-menu, 46, 112–119
- 'no-vscroll, 9, 14–16
- number-snip:make-fraction-snip, 254
- number-snip:make-repeating-decimal-snip, 254

- on-activate, 51, 100, 110
- on-change, 36, 137, 148, 160
- on-change-style, 137, 148, 160
- on-char, 137, 149, 160
- on-close, 27, 37, 51, 56, 58, 91, 98, 101, 110
- on-close-all, 122
- on-default-char, 137, 149, 160, 224
- on-default-event, 138, 149, 160
- on-delete, 138, 150, 161
- on-disable-surrogate, 24, 138, 161, 188
- on-display-size, 138, 150, 161, 223
- on-drop-file, 48
- on-edit-sequence, 30, 138, 150, 161, 210
- on-enable-surrogate, 24, 139, 161, 189
- on-event, 139, 151, 162
- on-exit, 48
- on-focus, 12, 23, 31, 139, 151, 162
- on-insert, 139, 152, 162, 195, 199
- on-load-file, 139, 152, 162, 210
- on-local-char, 140, 152, 162, 201, 223
- on-local-event, 140, 153, 163
- on-new-box, 31, 140, 153, 163
- on-new-image-snip, 140, 153, 163
- on-new-string-snip, 141, 154, 163
- on-new-tab-snip, 141, 154, 164
- on-paint, 141, 154, 164, 196, 210
- on-save-file, 37, 141, 155, 164, 214
- on-set-size-constraint, 23, 142, 155, 164
- on-size, 13, 50
- on-snip-modified, 142, 156, 165
- on-submit, 220
- on-subwindow-event, 172
- on-superwindow-show, 11, 49
- open-here, 99
- “open-line”, 252
- open-status-line, 52
- overwrite-status-changed, 57

- panel:dragable-mixin, 171
- panel:dragable<%>, 170
- panel:horizontal-dragable-mixin, 173
- panel:horizontal-dragable<%>, 171
- panel:horizontal-dragable%, 174
- panel:single-mixin, 167
- panel:single-pane%, 169

- panel:single-window-mixin, 168
- panel:single-window<%>, 168
- panel:single<%>, 167
- panel:single%, 169
- panel:vertical-dragable-mixin, 173
- panel:vertical-dragable<%>, 171
- panel:vertical-dragable%, 174
- parent, 9, 13–16, 46, 112–118, 131, 132, 169, 174
- “paste-click-region”, 253
- “paste-clipboard”, 252
- pasteboard:backup-autosave%, 176
- pasteboard:basic%, 175
- pasteboard:file%, 176
- pasteboard:info%, 176
- pasteboard:keymap%, 175
- pasteboard:standard-style-list%, 175
- path-utils:generate-autosave-name, 254
- path-utils:generate-backup-name, 255
- place-children, 168, 172
- preferences, 178
- preferences:add-callback, 255
- preferences:add-can-close-dialog-callback, 255
- preferences:add-editor-checkbox-panel, 255
- preferences:add-font-panel, 255
- preferences:add-on-close-dialog-callback, 256
- preferences:add-panel, 256
- preferences:add-scheme-checkbox-panel, 256
- preferences:add-to-editor-checkbox-panel, 256
- preferences:add-to-scheme-checkbox-panel, 257
- preferences:add-to-warnings-checkbox-panel, 257
- preferences:add-warnings-checkbox-panel, 257
- preferences:get, 257
- preferences:hide-dialog, 257
- preferences:restore-defaults, 257
- preferences:save, 258
- preferences:set, 258
- preferences:set-default, 258
- preferences:set-un/marshall, 258
- preferences:show-dialog, 259
- preferences:silent-save, 259
- put-file, 31

- quitting, 39

- recalc-snips, 13
- remove-autosave, 36

- remove-chained-keymap, 127
- remove-frame, 122
- remove-parens-forward, 185
- remove-sexp, 185
- “remove-space”, 252
- replace, 107
- replace-all, 107
- replace&search, 107
- reset-input-box, 220
- reset-region, 19
- ‘resize-corner, 9, 14–16
- resized, 211
- restore-keybinding, 130
- restore-underscores, 131
- return, 226
- revert, 93
- “ring-bell”, 252
- run-after-edit-sequence, 28

- ‘same, 37, 214
- save, 94
- save-as, 94
- “save-file”, 252
- “save-file-as”, 252
- save-file-out-of-date?, 28
- save-file/gui-error, 28
- scheme:add-coloring-preferences-panel, 259
- scheme:add-preferences-panel, 259
- scheme:get-color-prefs-table, 259
- scheme:get-keymap, 260
- scheme:get-wordbreak-map, 260
- scheme:init-wordbreak-map, 260
- scheme:set-mode-mixin, 189
- scheme:setup-keymap, 260
- scheme:sexp-snip<%>, 179
- scheme:sexp-snip%, 179
- scheme:short-sym->pref-name, 260
- scheme:short-sym->style-name, 260
- scheme:text-balanced?, 261
- scheme:text-mixin, 187
- scheme:text-mode-mixin, 188
- scheme:text-mode<%>, 188
- scheme:text-mode%, 190
- scheme:text<%>, 182
- scheme:text%, 189
- scrolls-per-page, 9, 13–16
- search-again, 107
- select-backward-sexp, 185
- “select-click-line”, 253
- “select-click-word”, 253
- select-down-sexp, 185
- select-forward-sexp, 186
- select-up-sexp, 186

- send-eof-to-box-in-port, 221
- send-eof-to-in-port, 221
- set-active-frame, 122
- set-allow-edits, 221
- set-anchor, 213
- set-delegate, 202
- set-editor, 12
- set-filename, 35
- set-info-canvas, 55
- set-insertion-point, 221
- set-label, 99
- set-label-prefix, 94
- set-macro-recording, 57
- set-modified, 37
- set-open-here-frame, 123
- set-overwrite-mode, 213
- set-percentages, 170
- set-search-direction, 107
- set-styles-fixed, 194
- set-surrogate, 133
- set-tab-size, 186
- set-unread-start-point, 221
- shortcut, 131, 132
- show, 49
- ‘show-caret, 196, 211
- show-delegated-text, 104
- ‘show-inactive-caret, 196, 211
- show-info, 55
- size-preferences-key, 112
- skip-whitespace, 19
- snip%, 179
- spacing, 46, 112–118, 169, 174
- split, 205, 207
- ‘standard, 37, 214
- start-colorer, 20
- “start-macro”, 253
- stop-colorer, 21
- stretchable-height, 9, 13–16, 46, 112–118, 169, 174
- stretchable-width, 9, 13–16, 46, 112–118, 169, 174
- string-snip%, 202
- style, 9, 13–16, 46, 112–118, 169, 174
- style lists
 - in an editor, 142, 195, 198, 214, 224
- submit-to-port?, 222

- tab-snip%, 205
- tab-stops, 23, 142, 189, 195, 198, 214, 224–228
- tabify, 186
- tabify-all, 186
- tabify-on-return?, 186
- tabify-selection, 187
- test.ss, 3

test:button-push, 261
test:close-top-level-window, 261
test:current-get-eventspaces, 261
test:keystroke, 262
test:menu-select, 262
test:mouse-click, 262
test:new-window, 263
test:number-pending-actions, 263
test:reraise-error, 263
test:run-interval, 263
test:run-one, 263
test:set-check-box!, 263
test:set-choice!, 264
test:set-radio-box!, 264
test:set-radio-box-item!, 264
test:top-level-focus-window-has?, 264
'text, 37, 214
'text-force-cr, 37, 214
text-mode%, 189
text:1-pixel-string-snip%, 202
text:1-pixel-tab-snip%, 205
text:autowrap%, 226
text:backup-autosave%, 227
text:basic-mixin, 195
text:basic<%>, 193
text:basic%, 224
text:clever-file-format-mixin, 213
text:clever-file-format<%>, 213
text:clever-file-format%, 227
text:delegate-mixin, 208
text:delegate<%>, 202
text:delegate%, 225
text:file-mixin, 215
text:file<%>, 214
text:file%, 227
text:foreground-color-mixin, 197
text:foreground-color<%>, 196
text:hide-caret/selection-mixin, 197
text:hide-caret/selection<%>, 197
text:hide-caret/selection%, 224
text:info-mixin, 212
text:info<%>, 211
text:info%, 228
text:input-box-mixin, 223
text:input-box<%>, 223
text:input-box%, 226
text:keymap%, 226
text:nbsp->space-mixin, 198
text:nbsp->space<%>, 198
text:nbsp->space%, 225
text:ports-mixin, 222
text:ports<%>, 216
text:ports%, 223
text:return-mixin, 200
text:return<%>, 200
text:return%, 226
text:searching-mixin, 200
text:searching<%>, 199
text:searching%, 227
text:standard-style-list%, 225
text:wide-snip-mixin, 201
text:wide-snip<%>, 201
text:wide-snip%, 225
thaw-colorer, 21
the-style-list, 179
"toggle-anchor", 252
"toggle-overwrite", 253
toggle-search-focus, 108
'toolbar-button, 46, 112–119
'transparent, 9, 14–16
"transpose-chars", 252
transpose-sexp, 187
"transpose-words", 252
uncomment-selection, 187
unhide-search, 108
up-sexp, 187
"upcase-word", 252
update-frame-filename, 34
update-info, 55, 58
update-region-end, 21
update-status-line, 52
version:add-spec, 265
version:version, 265
vert-margin, 9, 13–16, 169, 174
vertical-inset, 9, 13–16
wheel on mouse, 10
wheel-step, 9, 13–16
width, 46, 112–118
Windows menu, 46, 120
write, 181
x, 46, 112–118
y, 46, 112–118