PLT Framework: GUI Application Framework

Robert Bruce Findler
robb@cs.rice.edu

Matthew Flatt
mflatt@cs.utah.edu

Version 103
August 2000
Copyright notice

Copyright ©1996-2000 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.
6.8 canvas:info% = (canvas:info-mixin canvas:basic%) ................................. 15
6.9 canvas:wide-snippet% = (canvas:wide-snippet-mixin canvas:basic%) ......... 15

7 Editor
7.1 editor:basic<%> ................................................................. 17
7.2 editor:basic-mixin ............................................................ 18
7.3 editor:keymap<%> .............................................................. 21
7.4 editor:keymap-mixin ......................................................... 21
7.5 editor:autowrap<%> ........................................................... 22
7.6 editor:autowrap-mixin ....................................................... 22
7.7 editor:file<%> ................................................................. 22
7.8 editor:file-mixin .............................................................. 22
7.9 editor:backup-autosave<%> ................................................ 24
7.10 editor:backup-autosave-mixin ........................................... 25
7.11 editor:info<%> ............................................................... 26
7.12 editor:info-mixin ............................................................ 26

8 Exit
8.1 Exit Utilities ................................................................. 27

9 Exceptions ................................................................................. 29

10 Finder
10.1 Finder Utilities ................................................................. 30

11 Frame
11.1 frame:basic<%> ................................................................. 34
11.2 frame:basic-mixin ............................................................. 36
11.3 frame:info<%> ................................................................. 38
11.4 frame:info-mixin ............................................................. 39
11.5 frame:text-info<%> ........................................................... 40
11.6 frame:text-info-mixin ....................................................... 41
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.7</td>
<td>frame:pasteboard-info%</td>
<td>41</td>
</tr>
<tr>
<td>11.8</td>
<td>frame:pasteboard-info-mixin</td>
<td>42</td>
</tr>
<tr>
<td>11.9</td>
<td>frame:standard-menus%</td>
<td>42</td>
</tr>
<tr>
<td>11.10</td>
<td>frame:standard-menus-mixin</td>
<td>65</td>
</tr>
<tr>
<td>11.11</td>
<td>frame:editor%</td>
<td>66</td>
</tr>
<tr>
<td>11.12</td>
<td>frame:editor-mixin</td>
<td>68</td>
</tr>
<tr>
<td>11.13</td>
<td>frame:text%</td>
<td>70</td>
</tr>
<tr>
<td>11.14</td>
<td>frame:text-mixin</td>
<td>71</td>
</tr>
<tr>
<td>11.15</td>
<td>frame:pasteboard%</td>
<td>71</td>
</tr>
<tr>
<td>11.16</td>
<td>frame:pasteboard-mixin</td>
<td>71</td>
</tr>
<tr>
<td>11.17</td>
<td>frame:searchable%</td>
<td>72</td>
</tr>
<tr>
<td>11.18</td>
<td>frame:searchable-mixin</td>
<td>74</td>
</tr>
<tr>
<td>11.19</td>
<td>frame:file%</td>
<td>76</td>
</tr>
<tr>
<td>11.20</td>
<td>frame:file-mixin</td>
<td>76</td>
</tr>
<tr>
<td>11.21</td>
<td>frame:basic% = (frame:basic-mixin frame%)</td>
<td>77</td>
</tr>
<tr>
<td>11.22</td>
<td>frame:info% = (frame:info-mixin frame:basic%)</td>
<td>78</td>
</tr>
<tr>
<td>11.23</td>
<td>frame:text-info% = (frame:text-info-mixin frame:info%)</td>
<td>78</td>
</tr>
<tr>
<td>11.24</td>
<td>frame:pasteboard-info% = (frame:pasteboard-info-mixin frame:text-info%)</td>
<td>79</td>
</tr>
<tr>
<td>11.25</td>
<td>frame:standard-menus% = (frame:standard-menus-mixin frame:pasteboard-info%)</td>
<td>80</td>
</tr>
<tr>
<td>11.26</td>
<td>frame:editor% = (frame:editor-mixin frame:standard-menus%)</td>
<td>81</td>
</tr>
<tr>
<td>11.27</td>
<td>frame:text% = (frame:text-mixin frame:editor%)</td>
<td>82</td>
</tr>
<tr>
<td>11.28</td>
<td>frame:text-info-file% = (frame:file-mixin frame:text%)</td>
<td>83</td>
</tr>
<tr>
<td>11.29</td>
<td>frame:searchable% = (frame:searchable-mixin frame:text-info-file%)</td>
<td>83</td>
</tr>
<tr>
<td>11.30</td>
<td>frame:pasteboard% = (frame:pasteboard-mixin frame:editor%)</td>
<td>84</td>
</tr>
<tr>
<td>11.31</td>
<td>frame:pasteboard-info-file% = (frame:file-mixin frame:pasteboard%)</td>
<td>85</td>
</tr>
<tr>
<td>11.32</td>
<td>Frame Utilities</td>
<td>86</td>
</tr>
</tbody>
</table>

### 12 Group

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>group:%</td>
<td>87</td>
</tr>
<tr>
<td>12.2</td>
<td>Group Utilities</td>
<td>89</td>
</tr>
</tbody>
</table>
13 GUI Utilities 90
   13.1 gui-utils:text-snip<%> ............................... 90
   13.2 Gui-utils Utilities .................................... 90

14 Handler 93
   14.1 Handler Utilities ...................................... 93

15 Icon 95
   15.1 Icon Utilities .......................................... 95

16 Keymap 97
   16.1 keymap:aug-keymap<%> .................................. 97
   16.2 keymap:aug-keymap-mixin .............................. 97
   16.3 keymap:aug-keymap% = (keymap:aug-keymap-mixin keymap%) 100
   16.4 Keymap Utilities ........................................ 100

17 Keys 106
   17.1 Keys Utilities .......................................... 106

18 Main 107

19 Match Cache 108
   19.1 match-cache:% ......................................... 108

20 Panel 110
   20.1 panel:single<%> ......................................... 110
   20.2 panel:single-mixin ............................... 110
   20.3 panel:single-window<%> ............................... 111
   20.4 panel:single-window-mixin .............................. 111
   20.5 panel:single% = (panel:single-window-mixin (panel:single-mixin panel%)) 112
   20.6 panel:single-pane% = (panel:single-mixin pane%) 112

21 Parenthesis 113
   21.1 Paren Utilities ........................................ 113
28.3 text:searching% ................................................. 142
28.4 text:searching-mixin ........................................... 142
28.5 text:return% ...................................................... 142
28.6 text:return-mixin ............................................... 142
28.7 text:info% ......................................................... 143
28.8 text:info-mixin ................................................... 143
28.9 text:clever-file-format% ...................................... 145
28.10 text:clever-file-format-mixin ................................ 145
28.11 text:basic% = (text:basic-mixin (editor:basic-mixin text%)) ......... 146
28.12 text:keymap% = (editor:keymap-mixin text:basic%) ..................... 146
28.13 text:return% = (text:return-mixin text:keymap%) ....................... 147
28.14 text:autowrap% = (editor:autowrap-mixin text:keymap%) ................. 147
28.15 text:file% = (editor:file-mixin text:autowrap%) ......................... 147
28.16 text:clever-file-format% = (text:clever-file-format-mixin text:file%) ...... 148
28.17 text:backup-autosave% = (editor:backup-autosave-mixin text:clever-file-format%) 148
28.18 text:searching% = (text:searching-mixin text:backup-autosave%) ........ 148
28.19 text:info% = (text:info-mixin (editor:info-mixin text:searching%)) ......... 149

29 Version 150

29.1 Version Utilities ..................................................... 150

Index 151
Part I

Introduction
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*.

1.1 Thanks

Thanks to Shriram Krishnamurthi, Cormac Flanagan, Matthias Felleisen, 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{} and a patched version of latex2html. Some typesetting macros were originally taken from Julian Smart’s *Reference Manual for wxWindows 1.60: a portable C++ GUI toolkit*. 
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, section 11,
- Editors, section 7,
- and many others.

See the libraries, in 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-library "framework.ss" "framework") This library adds all of the definitions and macros in this manual to the top-level.
  
  - `(require-library "frameworkr.ss" "framework")
    This library adds a signature definitions framework^ and signature definitions for each of the constituent units for the "frameworkr.ss" unit. It also requires the macro.ss library, the tests.ss library.
  
  - `(require-library "frameworkr.ss" "framework")
    This library returns a unit/sig, §7.3.2 in PLT MzScheme: Language Manual that imports two units. The first must be the result of `(require-library "corer.ss")`, §15.2.6 in PLT MzScheme: Language Manual and the second must be mred@, which is built in to PLT MrEd: Graphical Toolbox Manual.
    This unit exports several sub-units. Each sub unit corresponds to one chapter in the Reference part of this manual.
  
  - `(require-library "macro.ss" "framework") This defines the mixin macro. See Mixins, section 3.2 for more information.

- **Test Suite Engine**
  
  - `(require-library "test.ss" "framework") This library adds all of the definitions in chapter 27 to the top-level.
  
  - `(require-library "tests.ss" "framework")
    This library adds all of the signature definitions needed for the testing engine to the top-level.
  
  - `(require-library "testr.ss" "framework")
    This library returns a unit/sig, §7.3.2 in PLT MzScheme: Language Manual that imports two units. The first must be mred@, which is built in to MrEd, and the second must be the result of `(require-library "keys.ss" "framework")

  - `(require-library "keys.ss" "framework")
    This library returns a unit/sig, §7.3.2 in PLT MzScheme: Language Manual that imports nothing. It matches the framework:keys^ signature, defined by the "tests.ss" library, described above.

- **GUI Utilities**
  
  - `(require-library "guiutils.ss" "framework")
    This libraries adds all of the definitions in chapter ?? to the top-level.
  
  - `(require-library "guiutilsr.ss" "framework")
    Requiring this library results in a unit that imports the mred@ unit and exports framework:gui-utils^.
  
  - `(require-library "guiutilss.ss" "framework")
3.2. Mixins

This libraries defines the signature `framework:gui-utils` that contains all of the names in chapter ???. These names do not have the `gui-utils:` prefix in the signature.

- **Preferences Library Filename Specification**

  This is an alternate library that allows you to specify the location of the preferences file for the preferences library, instead of using the default one. The expression:

  ```scheme
  (require-library "frameworkp.ss" "framework")
  ```

  evaluates to a unit that imports a unit matching the `mred` signature, one matching `mzlib:core` and one matching `framework:prefs-file`. The signature `framework:prefs-file` only contains one name: `preferences-filename`. This is expected to be a string naming the preferences file location.

The framework provides libraries to accommodate two styles of program construction: with units and through the top-level. Programmers who only use the top-level to construct their programs should use "framework.ss". Programmers who use units to construct their programs should first require "frameworks.ss" and then use "frameworkr.ss" to construct their programs.

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.rice.edu/CS/PLT/Publications/#ffk-pldi97](http://www.cs.rice.edu/CS/PLT/Publications/#ffk-pldi97). 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. It’s syntax is very similar to the syntax for `class`, §6 in [PLT MzScheme: Language Manual](http://www.cs.rice.edu/CS/PLT/Publications/#ffk-pldi97). The shape of a mixin is:

```scheme
(mixin (interface-expr ...) (interface-expr ...) initialization-variables
     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 initialization arguments specified by `initialization-arguments` and clauses specified by `instance-variable-clauses`. The syntax of the `initialization-variables` and `instance-variable-clause` are exactly the same as `class*/names`, §6.3 in [PLT MzScheme: Language Manual](http://www.cs.rice.edu/CS/PLT/Publications/#ffk-pldi97).

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:

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

The `mixin` macro is provided by

```scheme
(require-library "macro.ss" "framework")
```
3.3 Type Language

The types on method arguments are described using a slightly enhanced version of MrSpidey’s type language.

\[
T = \begin{cases} 
\text{rec } ([\langle \text{id} \rangle <T>[\ldots]) <T> & \text{recursive types} \\
\text{union } <T> \ldots & \text{union ”or”} \\
(T \ldots \rightarrow T \ldots) & \text{function (possible multiple value return)} \\
(T \ldots \rightarrow^{*} T) & \text{function – see below for explanation} \\
(T \rightarrow T \ldots) & \text{function – see below for explanation} \\
(T \rightarrow^{*} T) & \text{function – see below for explanation} \\
(\text{cons } T T) & \text{all pairs with components in } T \text{ and } T \\
(\text{vectorof } T) & \text{a vector whose elements are all } T \\
(\text{vector } T \ldots) & \text{a vector whose elements have types as listed} \\
\langle \text{symbol}\rangle & \text{‘a, ’b, etc.} \\
\langle \text{number}\rangle & \text{any scheme number} \\
\#t & \text{true} \\
\#f & \text{false} \\
\text{null} & \text{the empty list} \\
\text{TST} & \text{any value whatsoever} \\
\text{exact-int} & \text{a number that is both exact and an integer} \\
(\text{instance } c\% ) & \text{object created from } c\% \\
(\text{instance } i<\%>) & \text{object whose class implements } i<\%> \\
(\text{derived-from } c\%) & \text{class derived from } c\% \\
(\text{implements } i<\%>) & \text{class that implements } i<\%> \\
\langle \text{predicate}\rangle & \text{whatever value set the predicate recognizes} \\
\end{cases}
\]

\[
\text{predicate} = \begin{cases} 
\text{integer} & \text{integer} \\
\text{real} & \text{real} \\
\text{input-port} & \text{input-port} \\
\text{boolean} & \text{boolean} \\
\text{void} & \text{void} \\
\text{string} & \text{string} \\
\text{regexp} & \text{regexp} \\
\text{exact} & \text{exact} \\
\text{inexact} & \text{inexact} \\
\text{eof} & \text{eof} \\
\text{char} & \text{char} \\
\text{interface} & \text{interface} \\
\end{cases}
\]

The function types with asterisks are used for multiple arity and multiple return values. When an asterisk is present in front of the arrow, the type before the arrow must be a list and the elements of the list are the arguments of the function. Similarly, when an asterisk appears at the end of the arrow the type after the arrow must be a list whose elements are the types of the multiple return values of the function.

There are also a few shorthands the documentation uses:

\[
\text{(listof } <T>\text{)} = \text{(rec } ([X \text{ (union } (\text{cons } T X) \text{ null})]) X) \\
\text{printable} = \text{(rec } ([X \text{ (union } (\text{cons } X X) \\
\text{ (vectorof } X) \\
\text{ number boolean void string})]) \\
\text{X})
\]
Part II

Reference
4. Application

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

4.1 Application Utilities

application:current-app-name

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%).

- (application:current-app-name name) ⇒ void
  
  name : string
  
  Sets the name of the application to name

- (application:current-app-name) ⇒ string
  
  returns the current name of the application.
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.

5.1 Autosave Utilities

```ruby
autosave:register

- (autosave:register obj) ⇒ void
  obj : (object [do-autosave (-¿ void)])
```

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.
6. Canvas

```
(class->interface editor-canvas%)
  | canvas:basic<%>
  |  +--------------------------+
  |                    | canvas:info<%> canvas:wide-snip<%>
```

6.1 canvas:basic<%>

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

6.2 canvas:basic-mixin

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

Implements: canvas:basic<%>

- (make-object canvas:basic-mixin% parent editor style scrolls-per-page) ⇒ canvas:basic-mixin% object
  
  `parent` : frame%, dialog%, panel%, or pane% object
  `editor` = #f : text% or pasteboard% object or #f
  `style` = null : list of symbols in '(no-hscroll no-vscroll hide-hscroll hide-vscroll)
  `scrolls-per-page` = 100 : exact integer in [1, 10000]

The `style` list can contain the following flags:

- 'no-hscroll — disallows horizontal scrolling
- 'no-vscroll — disallows vertical scrolling
- 'hide-hscroll — allows horizontal scrolling, but hides the horizontal scrollbar
- 'hide-vscroll — allows vertical scrolling, but hides the vertical scrollbar

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 `scrollsPerPage` argument sets this value.

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

6.3 canvas:info

Extends: canvas:basic

6.4 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

  Enables or disables the caret in the display’s editor, if there is one.

  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 a editor-canvas, then the new editor is not installed into the canvas.

- (send a-canvas:info-mixin set-editor) ⇒ void

  If redraw? is #f, then the editor is not immediately drawn; in this case, something must force a redraw later (e.g., a call to the on-paint method).

  If the canvas has a line count installed with set-line-count, the canvas's minimum height is adjusted.

  Calls set-info-canvas to this canvas, if it has the focus in the frame.

6.5 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.
6.6. canvas:wide-snip-mixin

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%)

6.6 canvas:wide-snip-mixin

Domain: canvas:basic%

Implements: canvas:wide-snip%

Implements: canvas:basic%

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]

  If the canvas is displaying an editor, its on-display-size method is called.
  Adjusts the sizes of the marked snips.
  See add-wide-snip and add-tall-snip.

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

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

- (make-object canvas:basic% parent editor style scrolls-per-page) ⇒ 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-hscroll no-vscroll hide-hscroll hide-vscroll)
scrolls-per-page = 100 : exact integer in [1, 10000]

The style list can contain the following flags:
- ’no-hscroll — disallows horizontal scrolling
- ’no-vscroll — disallows vertical scrolling
- ’hide-hscroll — allows horizontal scrolling, but hides the horizontal scrollbar
- ’hide-vscroll — allows vertical scrolling, but hides the vertical scrollbar

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 scrollsPerPage argument sets this value.

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

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

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

- (make-object canvas:info% parent editor style scrolls-per-page) ⇒ 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-hscroll no-vscroll hide-hscroll hide-vscroll)
  scrolls-per-page = 100 : exact integer in [1, 10000]

The style list can contain the following flags:
- ’no-hscroll — disallows horizontal scrolling
- ’no-vscroll — disallows vertical scrolling
- ’hide-hscroll — allows horizontal scrolling, but hides the horizontal scrollbar
- ’hide-vscroll — allows vertical scrolling, but hides the vertical scrollbar

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 scrollsPerPage argument sets this value.

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

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

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

- (make-object canvas:wide-snip% parent editor style scrolls-per-page) ⇒ canvas:wide-snip% object
  parent : frame%, dialog%, panel%, or pane% object
  editor = #f : text% or pasteboard% object or #f
  style = null : list of symbols in ’(no-hscroll no-vscroll hide-hscroll hide-vscroll)
  scrolls-per-page = 100 : exact integer in [1, 10000]

The style list can contain the following flags:
- ’no-hscroll — disallows horizontal scrolling
- ’no-vscroll — disallows vertical scrolling
- ’hide-hscroll — allows horizontal scrolling, but hides the horizontal scrollbar
- ’hide-vscroll — allows vertical scrolling, but hides the vertical scrollbar
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 `scrollsPerPage` argument sets this value.

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

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

```
    editor<%>
     |   
    editor:basic<%> (class-<interface text<%>)
     |   
     +---------------------------+---------------------------+
     | editor:keymap<%> | editor:backup-autosave<%> | text:basic<%> | text:return<%> |
     | editor:autowrap<%> | editor:info<%> | text:clever-file-format<%> |
     +---------------------------+---------------------------+
         |   |   |   |   |   |   |
         | editor:file<%> |   |   |   | text:searching<%> |   | text:info<%> |
```

7.1 editor:basic<%>

Extends: editor<%>

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

**editing-this-file?**

Indicates if the file in this editor is open for editing, or merely for viewing.

- (send an-editor:basic editing-this-file?) ⇒ boolean

  Returns #f.

**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) ⇒ (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

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

locked?

Indicates if this buffer is locked. See also lock.

- (send an-editor:basic locked?) ⇒ boolean

on-close

This method is called when a frame that shows this buffer is closed.

- (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 : (→ 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.

7.2 editor:basic-mixin

Domain: editor<%>

Implements: editor:basic<%>
Implements: editor<%>

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

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.

begin-edit-sequence

The begin-edit-sequence and end-edit-sequence methods are used to bracket a set of editor modifications so that the results are all displayed at once. The commands may be nested arbitrarily deep. Using these functions can greatly speed up displaying the changes.

When a editor contains other editors, using begin-edit-sequence and end-edit-sequence on the main editor brackets some changes to the sub-editors as well, but it is not as effective when a sub-editor changes as calling begin-edit-sequence and end-edit-sequence for the sub-editor.

See also refresh-delayed?.

- (send an-editor:basic-mixin begin-edit-sequence undoable?) ⇒ void
  undoable? : boolean

  If the undoable? flag is #f, then the changes made in the sequence cannot be reversed through the undo method. This flag is only effective for the outermost begin-edit-sequence when nested sequences are used.

  See end-edit-sequence.

end-edit-sequence

See begin-edit-sequence.

- (send an-editor:basic-mixin end-edit-sequence) ⇒ void

  Unlike the primitive editor<%> method’s end-edit-sequence, this will raise an exception when it is called without an matching call to begin-edit-sequence.
get-file

Called when the user must be queried for a filename to load an editor. A starting directory string 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 : string 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.

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 section 8.8 (page 153).

- (send an-editor:basic-mixin lock lock?) ⇒ void
  lock? : boolean

  If lock? is #f, the editor is unlocked, otherwise it is locked.

  Maintains information for locked?.

on-edit-sequence

Called just before 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.

- (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 strings 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 : string or #f
  default-name : string 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.

7.3 editor:keymap<%>

Extends: editor:basic<%>

Classes matching this interface add support for mixing in multiple keymaps. They provide an extensible interface to chained keymaps, through the #mlink 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)

7.4 editor:keymap-mixin

Domain: editor:basic<%>

Implements: editor:basic<%>

Implements: editor:keymap<%>

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

Extends: editor:basic

Classes implementing this interface keep the auto-wrap state set based on the 'framework:auto-set-wrap? preference (see section ?? for more info about 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.

7.6  editor:autowrap-mixin

Domain: editor:basic

Implements: editor:basic

Implements: editor:autowrap

See editor:autowrap

on-close

This method is called when a frame that shows this buffer is closed.

- (send an-editor:autowrap-mixin on-close) ⇒ void

  Does nothing.

  Removes the preferences callback for 'framework:auto-set-wrap? (which updates the wrapping in this text). It chains to the superclass’s method.

7.7  editor:file

Extends: editor:keymap

Objects supporting this interface are expected to support files.

7.8  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.

**after-load-file**

Called just after the editor is loaded from a file. The argument to the method specifies whether the load was successful or not. See also `can-load-file?` and `on-load-file`.

- (send an-editor: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-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).

**after-save-file**

Called just after the editor is saved to a file. The argument to the method specifies whether the save was successful or not. See also `can-save-file?` and `on-save-file`.

- (send an-editor: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).

**editing-this-file?**

Indicates if the file in this editor is open for editing, or merely for viewing.

- (send an-editor:file-mixin editing-this-file?) ⇒ boolean
  returns #t.

**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%))
  Defaultly returns `(list keymap:get-global)`
  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`).

The filename of an editor can be changed by the system in response to file loading and saving method calls, and such changes do not go through this method; use \texttt{on-load-file} and \texttt{on-save-file} to monitor such filename changes.

- \(\text{(send an-editor:file-mixin set-filename name temp?) \Rightarrow void}\)
  \begin{verbatim}
  name : string
  temp? = #f : boolean
  \end{verbatim}
  Sets the filename to \texttt{name}. If \texttt{filename} is \#f or \texttt{temporary?} is a true value, then the user will still be prompted for a name on future calls to \texttt{save-file} and \texttt{load-file}.
  Updates the filename on each frame displaying this editor, for each frame that matches \texttt{frame:editor}.

7.9 \texttt{editor:backup-autosave} \%

Extends: \texttt{editor:basic} \%

Classes matching this interface support backup files and autosaving.

\texttt{autosave?}

Indicates weather this \texttt{editor} should be autosaved.

- \(\text{(send an-editor:backup-autosave autosave?) \Rightarrow boolean}\)
  \begin{verbatim}
  Returns #t.
  \end{verbatim}

\texttt{backup?}

Indicates weather this \texttt{editor} should be backed up.

- \(\text{(send an-editor:backup-autosave backup?) \Rightarrow boolean}\)
  \begin{verbatim}
  Returns #t.
  \end{verbatim}

\texttt{do-autosave}

This method is called to perform the autosaving. See also \texttt{autosave:register}

- \(\text{(send an-editor:backup-autosave do-autosave) \Rightarrow void}\)
  \begin{verbatim}
  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.
  \end{verbatim}

\texttt{remove-autosave}

This method removes the autosave file associated with this \texttt{editor}.

- \(\text{(send an-editor:backup-autosave remove-autosave) \Rightarrow void}\)
7.10 editor:backup-autosave-mixin

Domain: editor:basic

Implements: editor:backup-autosave

Implements: editor:basic

This mixin adds backup and autosave functionality to an editor.
The result of this mixin uses the same initialization arguments as the mixin’s argument.

on-change

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 any 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.

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

This method is called when a frame that shows this buffer is closed.

- (send an-editor:backup-autosave-mixin on-close) ⇒ void
  Does nothing.
  Deletes the autosave file and turns off autosaving.

on-save-file

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 : string
  format : symbol in '(guess standard text text-force-cr same copy)
  The filename argument is the name the file will be saved to. See load-file for information about format.
  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?*.

- (send an-editor:backup-autosave-mixin set-modified modified?) ⇒ void

  modified? : boolean

  Sets the modification state to *modified?*. If *modified?* is #f and the editor’s undo or redo stack contains a system-created undoer that resets the modified state (because the preceding undo or redo action puts the editor back to a state where the modification state was #f), the undoer is disabled.

  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.

---

### 7.11 editor:info<%>

Extends: editor:basic<%>

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

---

### 7.12 editor:info-mixin

Domain: editor:basic<%>

Implements: editor:basic<%>

Implements: editor:info<%>

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 section 8.8 (page 153).

- (send an-editor:info-mixin lock lock?) ⇒ void

  lock? : boolean

  If *lock?* is #f, the editor is unlocked, otherwise it is locked.

  Uses run-after-edit-sequence to call lock-status-changed.
8. 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.

8.1 Exit Utilities

`exit:can-exit?`

Calls the “can-callbacks”. See `exit:insert-can?-callback` for more information.

```
- (exit:can-exit?) ⇒ void
```

`exit:exit`

`exit:exit` performs three actions:

- If the preference `''framework:verify-exit`is not `#f`, it prompts the user about quitting (see section ?? for more info about preferences), and if the user sanctions quitting (or if the preference is `#f`), it
- invokes the exit-callbacks, with `exit:can-exit?`If none of the “can?” callbacks return `#f`, it invokes `exit:on-exit`and then `exit` (a nzscheme procedure).

```
- (exit:exit) ⇒ (void)
```

`exit:frame-exiting`

This is a parameter whose value is used as the parent of the "Are you sure you want to exit" dialog.
8.1. Exit Utilities

- `(exit:frame-exiting frame) ⇒ void`
  `frame : (union #f (instance frame%) (instance dialog%))`
  sets the value of the parameter to `frame`.

- `(exit:frame-exiting) ⇒ (union #f (instance frame%) (instance dialog%))`
  returns the current value of the parameter.

`exit:insert-can?-callback`

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-can?-callback callback) ⇒ void`
  `callback : (-¿ boolean)`

`exit:insert-on-callback`

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:insert-on-callback callback) ⇒ void`
  `callback : (-¿ void)`

`exit:on-exit`

Calls the “on-callbacks”. See `exit:insert-on-callback` for more information.

- `(exit:on-exit) ⇒ void`
9. Exceptions
10. 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 section ?? for more info about preferences) `framework:file-dialogs`. It it is `common`, the common platform-independent dialogs are used and if it it is `std`, the standard platform-specific dialogs are used.

The finder:common-get-file and finder:common-put-file dialogs keep a separate directory state (for starting in the same place as the previous dialog ended); this directory can be obtained and set with the procedure finder:current-find-file-directory.

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 mzlib's file utilities, section 15.2.10.

10.1 Finder Utilities

finder:common-get-file

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 10 for more information.

- (finder:common-get-file directory prompt filter filter-msg parent) ⇒ string or #f
  directory = null : string or null
  prompt = "Select File" : string
  filter = #f : a regular expression or #f
  filter-msg = "That filename does not have the right form." : string
  parent = null : top-level-window<%>

finder:common-get-file-list

This procedure queries the user for a list of filenames, using a platform-independent dialog box.

See section 10 for more information.
- (finder:common-get-file-list directory prompt filter filter-msg parent) ⇒ (union (listof string) #f)
  
  directory = null: (union null string)
  prompt = "Select File": string
  filter = #f: #f or a regular expression
  filter-msg = "That filename does not have the right form." : string
  parent = #f: (union #f (instance top-level-window<%>))

finder:common-put-file

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 10 for more information.

- (finder:common-put-file name directory replace? prompt filter filter-msg parent) ⇒ (union #f string)
  
  name = "Untitled": string
  directory = null: (union #f string)
  replace? = #f: bool
  prompt = "Select File": string
  filter = #f: (union #f regexp)
  filter-msg = "That filename does not have the right form." : string
  parent = (finder:dialog-parent-parameter): (union (instance top-level-window<%>)) #f

finder:current-find-file-directory

This is a parameter of the MzScheme manual mz:parameters that specifies the parent for any of the dialogs created with: finder:std-get-file, finder:std-put-file, finder:common-get-file-list, finder:common-get-file, or finder:common-put-file.

- (finder:current-find-file-directory) ⇒ (instance top-level-window<%>)
  
  Returns the current value of the parameter.

- (finder:current-find-file-directory top-level-window) ⇒ void
  
  top-level-window: (instance top-level-window<%>)
  
  Sets the parameter to top-level-window

finder:default-extension

This function is used to hold the state for the file dialogs under windows. The extension supplied will be used if the user does not supply an extension for the filename.

- (finder:default-extension ext) ⇒ void
  
  ext : string
  
  Sets the default extension to ext

- (finder:default-extension) ⇒ string
  
  Returns the default extension.
finder:dialog-parent-parameter


- (finder:dialog-parent-parameter) ⇒ (union (instance top-level-window<%>) #f)
  Returns the current value of the parameter.

finder:get-file

Queries the user for a filename.

- (finder:get-file directory prompt filter filter-msg parent) ⇒ (union string #f)
  directory = #f : (union string #f)
  prompt = "Select File" : string
  filter = #f : (union regexp #f)
  filter-msg = "That filename does not have the right form." : string
  parent = (finder:dialog-parent-parameter) : (union (instance top-level-window<%>) #f)
  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

Queries the user for a filename.

- (finder:put-file name directory replace? prompt filter filter-msg parent) ⇒ (union string #f)
  name = "Untitled" : string
  directory = #f : (union #f string)
  replace? = #f : bool
  prompt = "Select File" : string
  filter = #f : (union #f regexp)
  filter-msg = "That filename does not have the right form." : string
  parent = (finder:dialog-parent-parameter) : (union (instance top-level-window<%>) #f)
  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

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 10 for more information.

- (finder:std-get-file directory prompt filter filter-msg parent) ⇒ string or #f
  directory = #f : (union #f string)
  prompt = "Select File" : string
  filter = #f : (union regexp #f)
  filter-msg = "That filename does not have the right form." : string
  parent = (finder:dialog-parent-parameter) : (union (instance top-level-window<%>) #f)
finder:std-put-file

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 10 for more information.

- (finder:std-put-file name directory replace? prompt filter filter-msg parent) ⇒ string or #f
  name = "Untitled": string
directory = #f: (union #f string)
replace? = #f: bool
prompt = "Select File": string
filter = #f: (union #f regexp)
filter-msg = "That filename does not have the right form." : string
parent = (finder:dialog-parent-parameter): (union (instance top-level-window<%>) #f)
11. Frame

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

This is the interface hierarchy:

```
(class INTERFACE frame)
    frame:basic
        frame:standard-menus
        frame:info
            frame:editor
            frame:text-info
            frame:pasteboard-info
        frame:text
            frame:pasteboard
            frame:file
    frame:searchable
```

11.1 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)))]))
```
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:

...  
(rename [super-make-root-area-container make-root-area-container])
(private
  [status-panel #f])
(override
  [make-root-area-container
    (lambda (class parent)
      (set! status-panel
        (super-make-root-area-container
          vertical-panel% parent))
      (let* ([root (make-object class status-panel)])
        ; ... add other children to status-panel ...
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 : (instance (implements area-container<%>))
  parent : (instance (implements area-container<%>))

Calls make-object with class and parent.
```

### 11.2 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.

```
- (make-object frame:basic-mixin% label parent width height x y style) ⇒ frame:basic-mixin%
  object
  label : string
  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 \([0, 10000]\) or #f
  y = #f : exact integer in \([0, 10000]\) or #f
  style = null : list of symbols in ' (no-resize-border no-caption no-system-menu mdi-parent mdi-child)

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 wasagers), 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:
11.2. frame:basic-mixin

- `no-resize-border` — omits the resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- `no-caption` — omits the title bar for the frame (Windows)
- `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)

If the `mdi-child` style is specified, the parent must be a frame with the `mdi-parent` style, otherwise an exn:application:mismatch 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.

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-close?

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:basic-mixin can-close?)` ⇒ bool
  Calls the method can-remove-frame? of group:get-the-frame-group.

on-close

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:basic-mixin on-close)` ⇒ void
  calls the remove-frame method of the result of group:get-the-frame-group.

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 MacOS, when the user double-clicks on a MrEd file or drags a file onto the MrEd icon, the on-drop-file method of the frontmost frame is called (if drag-and-drop is enabled for that frame).
11.3. frame:info<%>

Extends: frame:basic<%>

Frames matching this interface support a status line.

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%)

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%)

update-info

This method updates all of the information in the panel.

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

11.4 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:

  ...  
  (rename [super-make-root-area-container make-root-area-container])
  (private
[status-panel #f])
(override
[make-root-area-container
(lambda (class parent)
  (set! status-panel
    (super-make-root-area-container
     vertical-panel% parent))
  (let* ([root (make-object class 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%))

Calls make-object with class and parent.
Builds an extra panel for displaying various information.

on-close
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.

11.5 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

11.6 frame:text-info-mixin

Domain: frame:info

Implements: frame:text-info

Implements: frame:info

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

on-close

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 the preferences section 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.

11.7 frame:pasteboard-info

Extends: frame:info
11.8  frame:pasteboard-info-mixin

Domain:  frame:basic

Implements:  frame:pasteboard-info

Implements:  frame:basic

11.9  frame:standard_menus

Extends:  frame:basic

Classes matching this interface provides a skeleton for the standard set of menus in a frame.

The available methods, listed in order they appear in the menus, is:

- file-menu:between-new-and-open
- file-menu:between-open-and-revert
- file-menu:between-revert-and-save
- file-menu:between-save-as-and-print
- file-menu:between-print-and-close
- file-menu:between-close-and-quit
- file-menu:after-quit
- edit-menu:between-redo-and-cut
- edit-menu:between-cut-and-copy
- edit-menu:between-copy-and-paste
- edit-menu:between-paste-and-clear
- edit-menu:between-clear-and-select-all
- edit-menu:between-select-all-and-find
- edit-menu:between-find-and-preferences
- edit-menu:after-preferences
- help-menu:before-about
- 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 menu item.

dump-menu:between-clear-and-select-all

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.
11. Frame

11.9. frame:standard-menus

- (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

This method is called when the clear menu-item of the edit-menu menu is selected. If edit-menu:clear is bound to #f instead of a procedure, the clear menu item will not be created.

- (send a-frame:standard-menus edit-menu:clear item evt) ⇒ void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%)
  Defaultly bound to:
  
  (lambda (menu evt) (let ((edit (get-edit-target-object))) (when (and edit (is-a? edit editor<%>)

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 "Clear the selection without affecting paste"

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:
edit-menu:clear-string

The result of this method is used to construct the name of this menu. It is inserted between "(if (eq? (system-type) (quote macos)) Clear &Delete)" and "" to form the complete name.

- (send a-frame:standard-menus edit-menu:clear-string) ⇒ string

edit-menu:copy

This method is called when the copy menu-item of the edit-menu menu is selected. If edit-menu:copy is bound to #f instead of a procedure, the copy menu item will not be created.

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

Defaultly bound to:

  (lambda (item evt) (let ((editor (get-edit-target-object))) (when (and editor (is-a? editor editor<%>))

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 "Copy the selection"

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:

  (lambda (item) (let* ((editor (get-edit-target-object)) (enable? (and editor (is-a? editor editor<%>))

edit-menu:copy-string

The result of this method is used to construct the name of this menu. It is inserted between "&Copy" and "" to form the complete name.

- (send a-frame:standard-menus edit-menu:copy-string) ⇒ string
edit-menu:cut

This method is called when the cut menu-item of the edit-menu menu is selected. If edit-menu:cut is bound to #f instead of a procedure, the cut menu item will not be created.

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

Defaultly bound to:

(lambda (menu evt) (let ((edit (get-edit-target-object))) (when (and edit (is-a? editor<%>))...))

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 "Cut the selection"

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:

(lambda (item) (let* ((editor (get-edit-target-object)) (enable? (and editor (is-a? editor<%>))...))

edit-menu:cut-string

The result of this method is used to construct the name of this menu. It is inserted between "Cut" and "" to form the complete name

- (send a-frame:standard-menus edit-menu:cut-string) ⇒ string

edit-menu:find

This method is called when the find menu-item of the edit-menu menu is selected. If edit-menu:find is bound to #f instead of a procedure, the find menu item will not be created.

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

Defaultly bound to:

#f
11.9. frame:standard-menus

**edit-menu:find-again**

This method is called when the find-again menu-item of the edit-menu menu is selected. If edit-menu:find-again is bound to `#f` instead of a procedure, the find-again menu item will not be created.

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

Defaultly bound to:

`#f`
```

**edit-menu:find-again-help-string**

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

```scheme
- (send a-frame:standard-menus edit-menu:find-again-help-string) ⇒ string

Defaultly returns "Search for the same string as before"
```

**edit-menu:find-again-on-demand**

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

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

Defaultly is this:

(lamba (item) (send item enable (let ((target (get-edit-target-object))) (and target (is-a? target edit-menu:find-again-string)))))
```

**edit-menu:find-again-string**

The result of this method is used to construct the name of this menu. It is inserted between "Find Again" and "" to form the complete name

```scheme
- (send a-frame:standard-menus edit-menu:find-again-string) ⇒ string
```

**edit-menu:find-help-string**

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

```scheme
- (send a-frame:standard-menus edit-menu:find-help-string) ⇒ string

Defaultly returns "Search for a string in the window"
```

**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:

  (lambda (item) (send item enable (let ((target (get-edit-target-object))) (and target (is-a? target edit-menu:find-string)))) (and target (is-a? target edit-menu:find-string))

**edit-menu:find-string**

The result of this method is used to construct the name of this menu. It is inserted between "Find" and ". . ." to form the complete name

- (send a-frame:standard-menus edit-menu:find-string) ⇒ string

**edit-menu:get-clear-item**

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

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

**edit-menu:get-copy-item**

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

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

**edit-menu:get-cut-item**

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

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

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

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

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

**edit-menu:get-find-item**

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

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

**edit-menu:get-paste-item**

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

- (send a-frame:standard-menus edit-menu:get-paste-item) ⇒ (instance iscmclasmenu-item)
edit-menu:get-preferences-item

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

- (send a-frame:standard-menus edit-menu:get-preferences-item) ⇒ (instance iscmclasmenu-item)

edit-menu:get-redo-item

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

- (send a-frame:standard-menus edit-menu:get-redo-item) ⇒ (instance iscmclasmenu-item)

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

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

- (send a-frame:standard-menus edit-menu:get-replace-and-find-again-item) ⇒ (instance iscmclasmenu-item)

edit-menu:get-select-all-item

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

- (send a-frame:standard-menus edit-menu:get-select-all-item) ⇒ (instance iscmclasmenu-item)

edit-menu:get-undo-item

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

- (send a-frame:standard-menus edit-menu:get-undo-item) ⇒ (instance iscmclasmenu-item)

edit-menu:paste

This method is called when the paste menu-item of the edit-menu menu is selected. If edit-menu:paste is bound to #f instead of a procedure, the paste menu item will not be created.

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

Defaultly bound to:

(lambda (menu evt) (let ((edit (get-edit-target-object))) (when (and edit (is-a? edit editor%))

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 "Paste the most recent copy or cut over the selection"
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:

  (lambda (item) (let* ((editor (get-edit-target-object)) (enable? (and editor (is-a? editor edit-menu:paste-string))) )
  ⇒

edit-menu:paste-string

The result of this method is used to construct the name of this menu. It is inserted between "&Paste" and "" to form the complete name

- (send a-frame:standard-menus edit-menu:paste-string) ⇒ string

edit-menu:preferences

This method is called when the preferences menu-item of the edit-menu menu is selected. If edit-menu:preferences is bound to #f instead of a procedure, the preferences menu item will not be created.

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

  Defaultly bound to:

  (lambda (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 "Configure the preferences"

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:

  void
edit-menu:preferences-string

The result of this method is used to construct the name of this menu. It is inserted between "Preferences..." and "" to form the complete name.

- (send a-frame:standard-menus edit-menu:preferences-string) ⇒ string

edit-menu:redo

This method is called when the redo menu-item of the edit-menu menu is selected. If edit-menu: redo is bound to #f instead of a procedure, the redo menu item will not be created.

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

Defaultly bound to:

  (lambda (menu evt) (let ((edit (get-edit-target-object))) (when (and edit (is-a? editor<%>)

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

Defaultly returns "Redo the most recent undo"

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%

Defaultly is this:

  (lambda (item) (let* ((editor (get-edit-target-object)) (enable? (and editor (is-a? editor<%>)

edit-menu: redo-string

The result of this method is used to construct the name of this menu. It is inserted between "&Redo" and "" to form the complete name

- (send a-frame:standard-menus edit-menu: redo-string) ⇒ string

edit-menu: replace-and-find-again

This method is called when the replace-and-find-again menu-item of the edit-menu menu is selected. If edit-menu:replace-and-find-again is bound to #f instead of a procedure, the replace-and-find-again menu item will not be created.
11. Frame

11.9. frame:standard-menus

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

Defaultly bound to:

#f

default: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 "Replace the current text and search for the same string as before"

default: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:

(lambda (item) (send item enable (let ((target (get-edit-target-object))) (and target (is-a? target editor%)))))

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

The result of this method is used to construct the name of this menu. It is inserted between "Replace && Find Again" and "" to form the complete name.

- (send a-frame:standard-menus edit-menu:replace-and-find-again-string) ⇒ string

default:edit-menu:select-all

This method is called when the select-all menu-item of the edit-menu menu is selected. If edit-menu:select-all is bound to #f instead of a procedure, the select-all menu item will not be created.

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

Defaultly bound to:

(lambda (menu evt) (let ((edit (get-edit-target-object))) (when (and edit (is-a? edit editor%)))))

default: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

Defaultly returns "Select the entire document"
edit-menu:select-all-on-demand

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

\[- (send \ a-frame:standard-menus \ edit-menu:select-all-on-demand \ item) \Rightarrow \text{void} \]

Defaultly is this:

\[(\lambda (item) \ (\text{let}\* \ ((\text{editor} \ (\text{get-edit-target-object}))) \ (\text{enable}?) \ (\text{and} \ \text{editor} \ (\text{is-a?} \ \text{editor} \%))) \Rightarrow \text{void})\]

edit-menu:select-all-string

The result of this method is used to construct the name of this menu. It is inserted between ”Select All” and ”” to form the complete name

\[- (send \ a-frame:standard-menus \ edit-menu:select-all-string) \Rightarrow \text{string} \]

edit-menu:undo

This method is called when the undo menu-item of the edit-menu menu is selected. If edit-menu:undo is bound to #f instead of a procedure, the undo menu item will not be created.

\[- (send \ a-frame:standard-menus \ edit-menu:undo \ item \ evt) \Rightarrow \text{void} \]

\[item : (\text{instance} \ (\text{derived-from} \ \text{menu-item}%)) \]

\[evt : (\text{instance} \ \text{control-event}%)\]

Defaultly bound to:

\[(\lambda (\text{menu} \ \text{evt}) \ (\text{let} \ ((\text{edit} \ (\text{get-edit-target-object}))) \ (\text{when} \ (\text{and} \ \text{edit} \ (\text{is-a?} \ \text{edit} \%))) \Rightarrow \text{void}))\]

edit-menu:undo-help-string

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

\[- (send \ a-frame:standard-menus \ edit-menu:undo-help-string) \Rightarrow \text{string} \]

Defaultly returns ”Undo the most recent action”

edit-menu:undo-on-demand

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

\[- (send \ a-frame:standard-menus \ edit-menu:undo-on-demand \ item) \Rightarrow \text{void} \]

Defaultly is this:

\[(\lambda (item) \ (\text{let}\* \ ((\text{editor} \ (\text{get-edit-target-object}))) \ (\text{enable}?) \ (\text{and} \ \text{editor} \ (\text{is-a?} \ \text{editor} \%))) \Rightarrow \text{void})\]
edit-menu:undo-string

The result of this method is used to construct the name of this menu. It is inserted between "&Undo" and "" to form the complete name

- (send a-frame:standard-menus edit-menu:undo-string) ⇒ string

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%))
  Adds a separator menu item.

file-menu:close

This method is called when the close menu-item of the file-menu menu is selected. If file-menu:close is bound to #f instead of a procedure, the close menu item will not be created.

- (send a-frame:standard-menus file-menu:close item evt) ⇒ void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%)
  Defaultly bound to:
    (lambda (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 "Close this file"

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:
void

file-menu:close-string

The result of this method is used to construct the name of this menu. It is inserted between "&Close" and "" to form the complete name

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

file-menu:get-close-item

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

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

file-menu:get-new-item

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

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

file-menu:get-open-item

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

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

file-menu:get-print-item

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

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

file-menu:get-quit-item

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

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

file-menu:get-revert-item

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

- (send a-frame:standard-menus file-menu:get-revert-item) ⇒ (instance iscmclasmenu-item)
11.9. frame:standard-menus<%>

file-menu:get-save-as-item

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

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

file-menu:get-save-item

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

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

file-menu:new

This method is called when the new menu-item of the file-menu menu is selected. If file-menu:new is bound to #f instead of a procedure, the new menu item will not be created.

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

  (lambda (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 "Open a new file"

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:

  void

file-menu:new-string

The result of this method is used to construct the name of this menu. It is inserted between "&New" and "" to form the complete name

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

This method is called when the open menu-item of the file-menu menu is selected. If file-menu:open is bound to #f instead of a procedure, the open menu item will not be created.

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

Defaultly bound to:

  (lambda (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

Defaultly returns "Open a file from disk"

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%

Defaultly is this:

    void

file-menu:open-string

The result of this method is used to construct the name of this menu. It is inserted between "&Open" and "..." to form the complete name

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

file-menu:print

This method is called when the print menu-item of the file-menu menu is selected. If file-menu:print is bound to #f instead of a procedure, the print menu item will not be created.

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

Defaultly bound to:

    #f
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 "Print this file"

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:

    void

file-menu:print-string

The result of this method is used to construct the name of this menu. It is inserted between "&Print" and "..." to form the complete name

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

file-menu:quit

This method is called when the quit menu-item of the file-menu menu is selected. If file-menu:quit is bound to #f instead of a procedure, the quit menu item will not be created.

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

    (lambda (item control) (parameterize ((exit:frame-exiting this)) (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
  Defaultly returns "Quit"

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%
  Defaultly is this:

  void

file-menu:quit-string

The result of this method is used to construct the name of this menu. It is inserted between "(if (eq? (system-type) (quote windows)) E&xit Quit)" and "" to form the complete name

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

file-menu:revert

This method is called when the revert menu-item of the file-menu menu is selected. If file-menu:revert is bound to #f instead of a procedure, the revert menu item will not be created.

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

  #f

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
  Defaultly returns "Revert this file to the copy on disk"

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:

  void

file-menu:revert-string

The result of this method is used to construct the name of this menu. It is inserted between "&Revert" and "" to form the complete name

- (send a-frame:standard-menus file-menu:revert-string) ⇒ string
file-menu:save

This method is called when the save menu-item of the file-menu menu is selected. If file-menu:save is bound to #f instead of a procedure, the save menu item will not be created.

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

Defaultly bound to:

#f
```

file-menu:save-as

This method is called when the save-as menu-item of the file-menu menu is selected. If file-menu:save-as is bound to #f instead of a procedure, the save-as menu item will not be created.

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

Defaultly bound to:

#f
```

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 "Prompt for a filename and save this file to disk"
```

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:

void
```

file-menu:save-as-string

The result of this method is used to construct the name of this menu. It is inserted between "Save" and "&As..." to form the complete name.

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

  Defaultly returns "Save this file to disk"

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%

  Defaultly is this:

    void

file-menu:save-string

The result of this method is used to construct the name of this menu. It is inserted between "&Save" and "" to form the complete name

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

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-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 class (see frame:standard-menus% for a list).

- (send a-frame:standard-menus get-menu-item%) ⇒ (derived-from menu-item%)  
  defaulty returns 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%)  
  defaulty returns menu%

help-menu:about

This method is called when the about menu-item of the help-menu menu is selected. If help-menu:about is bound to #f instead of a procedure, the about menu item will not be created.

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

Defaulty bound to:

  #f

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  
  Defaulty returns "Learn something about this application"

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%  

Defaulty is this:

  void
help-menu:about-string

The result of this method is used to construct the name of this menu. It is inserted between "About " and "." to form the complete name.

- (send a-frame:standard-menus help-menu:about-string) ⇒ string

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:get-about-item

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

- (send a-frame:standard-menus help-menu:get-about-item) ⇒ (instance iscmclasmenu-item)

11.10 frame:standard-menus-mixin

Domain: frame:basic<%>

Implements: frame:standard-menus<%>

Implements: frame:basic<%>

This frame provides a skeleton for the standard set of menus in a frame.

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

on-subwindow-char

Called when this window or a child window receives a keyboard event. The on-subwindow-char method of the receiver’s top-level window is called first (see get-top-level-window); if the return value is #f, then
the on-subwindow-char method is called for the next child in the path to the receiver, and so on. Finally, if the receiver’s on-subwindow-char method returns #f, the event is passed on to the receiver’s normal key-handling mechanism.

BEWARE: The default on-subwindow-char in frame% and on-subwindow-char in dialog% methods consume certain keyboard events (e.g., arrow keys, Enter) used for navigating within the window. Because the top-level window gets the first chance to handle the keyboard event, some events never reach the “receiver” child unless the default frame or dialog method is overridden.

- (send a-frame:standard-menus-mixin on-subwindow-char receiver event) ⇒ bool
  receiver : window% object
  event : key-event% object

Returns the result of

  (or (send this on-menu-char event)
       (send this on-system-menu-char event)
       (send this on-traverse-char event))

If the 'framework:menu-bindings preference is true, returns the result of calling on-traverse-char, which effectively skips on-menu-char and on-system-menu-char. If the preference is false, this method chains to the superclass method, instead.

11.11 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%).

save-as

Queries the use for a file name and saves the file with that name.

- (send a-frame:editor save-as format) ⇒ void
  format = 'same : (union 'guess 'standard 'text 'text-force-cr 'same 'copy)
set-label-prefix

Sets the prefix for the label of the frame.

- (send a-frame:editor set-label-prefix prefix) ⇒ void
  prefix : string

11.12 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, an 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.

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%)


file-menu:print

This method is called when the print menu-item of the file-menu menu is selected. If file-menu:print is bound to #f instead of a procedure, the print menu item will not be created.

- (send a-frame:editor-mixin file-menu:print 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

This method is called when the revert menu-item of the file-menu menu is selected. If file-menu:revert is bound to #f instead of a procedure, the revert menu item will not be created.
- (send a-frame:editor-mixin file-menu:revert item evt) ⇒ void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%)

  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.

file-menu:save

This method is called when the save menu-item of the file-menu menu is selected. If file-menu:save is bound to #f instead of a procedure, the save menu item will not be created.

- (send a-frame:editor-mixin file-menu:save item evt) ⇒ void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%)

  Saves the file in the editor.

file-menu:save-as

This method is called when the save-as menu-item of the file-menu menu is selected. If file-menu:save-as is bound to #f instead of a procedure, the save-as menu item will not be created.

- (send a-frame:editor-mixin file-menu:save-as 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.

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. 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. Buttons and check boxes can have bitmap labels (only when they are created with bitmap labels), but all other windows have string labels.

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

This method is called when the about menu-item of the help-menu menu is selected. If help-menu:about is bound to #f instead of a procedure, the about menu item will not be created.

- (send a-frame:editor-mixin help-menu:about 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 used to construct the name of this menu. It is inserted between "About " and "..." to form the complete name

- (send a-frame:editor-mixin help-menu:about-string) ⇒ string
  
  Returns the result of (application:current-app-name)

on-close

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 or #f

  If l is #f, the window's label is removed.

  Sets the label, but preserve's the label's prefix. See also set-label-prefix.

11.13  frame:text<%>

Extends: frame:editor<%>

Frames matching this interface provide support for text%'s.
11.14  frame:text-mixin

Domain: frame:editor

Implements: frame:editor

Implements: frame:editor

This mixin 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.

11.15  frame:pasteboard

Extends: frame:editor

Frames matching this interface provide support for pasteboards.

11.16  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 \texttt{make-editor} to check that get-editor\% is returning a reasonable editor.

\[- (\texttt{send a-frame:pasteboard-mixin get-editor\%}) \Rightarrow \text{interface} \]

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.

\[- (\texttt{send a-frame:pasteboard-mixin get-editor\%}) \Rightarrow (\text{implements editor\%}) \]

Returns pasteboard:keymap\%.

11.17 frame:searchable\%

Extends: frame:text\%

Frames that implement this interface support searching.

get-text-to-search

Override this method to specify which text to search.

\[- (\texttt{send a-frame:searchable get-text-to-search}) \Rightarrow (\text{instance (derived-from text\%)}) \]

Returns the result of get-editor.

hide-search

This method hides the searching information on the bottom of the frame.

\[- (\texttt{send a-frame:searchable hide-search}) \Rightarrow \text{void} \]

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 \texttt{void} if the focus was not to the search text, otherwise it returns a boolean indicating the success of the search.

\[- (\texttt{send a-frame:searchable move-to-search-or-reverse-search}) \Rightarrow (\text{union boolean void}) \]
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.
11.18. frame:searchable-mixin

Domain: frame:text

Implements: frame:text

Implements: frame:searchable

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:find

This method is called when the find menu-item of the edit-menu menu is selected. If edit-menu:find is bound to #f instead of a procedure, the find menu item will not be created.

- (send a-frame:searchable-mixin edit-menu:find item evt) ⇒ void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%)
  Calls move-to-search-or-search.

edit-menu:find-again

This method is called when the find-again menu-item of the edit-menu menu is selected. If edit-menu:find-again is bound to #f instead of a procedure, the find-again menu item will not be created.

- (send a-frame:searchable-mixin edit-menu:find-again) ⇒ boolean
  Returns #t, and searches for the same text that was last searched for in the text.

edit-menu:replace-and-find-again

This method is called when the replace-and-find-again menu-item of the edit-menu menu is selected. If edit-menu:replace-and-find-again is bound to #f instead of a procedure, the replace-and-find-again menu item
will not be created.

- (send a-frame:searchable-mixin edit-menu:replace-and-find-again) ⇒ 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.

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-mixin get-editor<%>) ⇒ interface
  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-mixin get-editor%) ⇒ (implements editor<%>)
  This method returns text:searching%.

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:

... (rename [super-make-root-area-container make-root-area-container]) (private
  [status-panel #f]) (override
    [make-root-area-container
      (lambda (class parent)
        (set! status-panel
          (super-make-root-area-container vertical-panel% parent))
          (let* ([root (make-object class 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<%>)

Calls make-object with class and parent.

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.

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**

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.

11.19  frame:file<%>

Extends: frame:editor<%>

Frames supporting this interface manage editors, like frames supporting the frame:editor<%> interface, but in addition, they support editors that match the editor:file<%> interface.

11.20  frame:file-mixin

Domain: frame:editor<%>

Implements: frame:file<%>

Implements: frame:editor<%>

This mixin adds support for editor:file<%> objects.

The result of this mixin uses the same initialization arguments as the mixin’s argument.
can-close?

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:file-mixin can-close?) ⇒ boolean
  Checks to see if the editor has been saved.

11.21 frame:basic% = (frame:basic-mixin frame%)

frame:basic% = (frame:basic-mixin frame%)
- (make-object frame:basic% label parent width height x y style) ⇒ frame:basic% object
  label : string
  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 [0, 10000] or #f
  y = #f : exact integer in [0, 10000] or #f
  style = null : list of symbols in '{no-resize-border no-caption no-system-menu mdi-parent mdi-child}

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 resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- 'no-caption — omits the title bar for the frame (Windows)
- '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)

If the 'mdi-child style is specified, the parent must be a frame with the 'mdi-parent style, otherwise an exn:application:mismatch 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.
11.22 frame:info% = (frame:info-mixin frame:basic%) 

frame:info% = (frame:info-mixin frame:basic%) 

- (make-object frame:info% label parent width height x y style) ⇒ frame:info% object
  
  label : string
  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 [0, 10000] or #f
  y = #f : exact integer in [0, 10000] or #f
  style = null : list of symbols in '(no-resize-border no-caption no-system-menu mdi-parent mdi-child)

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 resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- 'no-caption — omits the title bar for the frame (Windows)
- '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)

If the 'mdi-child style is specified, the parent must be a frame with the 'mdi-parent style, otherwise an exn:application:mismatch 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.

11.23 frame:text-info% = (frame:text-info-mixin frame:info%) 

frame:text-info% = (frame:text-info-mixin frame:info%) 

- (make-object frame:text-info% label parent width height x y style) ⇒ frame:text-info% object
  
  label : string
  parent = #f : frame% object or #f
  width = #f : exact integer in [0, 10000] or #f
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 resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- `no-caption` — omits the title bar for the frame (Windows)
- `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)

If the `mdi-child` style is specified, the `parent` must be a frame with the `mdi-parent` style, otherwise an `exn:application:mismatch` 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.

### 11.24 frame:pasteboard-info% = (frame:pasteboard-info-mixin frame:text-info%)

- `(make-object frame:pasteboard-info% label parent width height x y style) ⇒ frame:pasteboard-info%` object

  - `label` : string
  - `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 `[0, 10000]` or `#f`
  - `y` = `#f` : exact integer in `[0, 10000]` or `#f`
  - `style` = `null` : list of symbols in ’(no-resize-border no-caption no-system-menu mdi-parent mdi-child)

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 resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- `'no-caption` — omits the title bar for the frame (Windows)
- `'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)

If the `'mdi-child` style is specified, the `parent` must be a frame with the `'mdi-parent` style, otherwise an `exn:application:mismatch` 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.
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 resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- `no-caption` — omits the title bar for the frame (Windows)
- `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)

If the `mdi-child` style is specified, the parent must be a frame with the `mdi-parent` style, otherwise an `exn:application:mismatch` 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.

\[ \text{11.26 \ frame:editor\%} = (\text{frame:editor-mixin frame:standard-menus\%}) \]

\[
\begin{align*}
\text{frame:editor\%} & = (\text{frame:editor-mixin frame:standard-menus\%}) \\
\text{ - (make-object frame:editor\% label parent width height x y style) } & \rightarrow \text{ frame:editor\% object} \\
& \quad \text{label : string}
& \quad \text{parent = \#f : frame\% object or \#f}
& \quad \text{width = \#f : exact integer in \([0, 10000]\) or \#f}
& \quad \text{height = \#f : exact integer in \([0, 10000]\) or \#f}
& \quad \text{x = \#f : exact integer in \([0, 10000]\) or \#f}
& \quad \text{y = \#f : exact integer in \([0, 10000]\) or \#f}
& \quad \text{style = null : list of symbols in ' (no-resize-border no-caption no-system-menu mdi-parent mdi-child) }
\end{align*}
\]

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 resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- `no-caption` — omits the title bar for the frame (Windows)
11.27. frame:text% = (frame:text-mixin frame:editor%)

frame:text% = (frame:text-mixin frame:editor%)

- (make-object frame:text% label parent width height x y style) ⇒ frame:text% object
  label : string
  parent = #: frame% object or #f
  width = #: exact integer in [0, 10000] or #f
  height = #: exact integer in [0, 10000] or #f
  x = #: exact integer in [0, 10000] or #f
  y = #: exact integer in [0, 10000] or #f
  style = null : list of symbols in 'no-resize-border no-caption no-system-menu mdi-parent
                                            mdi-child"

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 resizeable border around the window (Windows) or grow box
    in the bottom right corner (MacOS)
  - 'no-caption — omits the title bar for the frame (Windows)
  - '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)

If the 'mdi-child style is specified, the parent must be a frame with the 'mdi-parent style, otherwise
an exn:application:mismatch 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.
11. Frame

11.28 \texttt{frame:text-info-file\%} = (frame:file-mixin frame:text\%)

\texttt{frame:text-info-file\%} = (frame:file-mixin frame:text\%)

- (make-object frame:text-info-file\% label parent width height x y style) \Rightarrow \texttt{frame:text-info-file\%}

  \texttt{object}

  \texttt{label : string}

  \texttt{parent = \#f : frame\% object or \#f}

  \texttt{width = \#f : exact integer in \([0, 10000]\) or \#f}

  \texttt{height = \#f : exact integer in \([0, 10000]\) or \#f}

  \texttt{x = \#f : exact integer in \([0, 10000]\) or \#f}

  \texttt{y = \#f : exact integer in \([0, 10000]\) or \#f}

  \texttt{style = null : list of symbols in 'no-resize-border no-caption no-system-menu mdi-parent mdi-child}

The \texttt{label} string is displayed in the frame’s title bar. If the frame’s label is changed (see \texttt{set-label}), the title bar is updated.

The \texttt{parent} argument can be \#f or an existing frame. Under Windows, if \texttt{parent} is an existing frame, the new frame is always on top of its parent. Also, the \texttt{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 \texttt{parent} is \#f, then the eventspace for the new frame is the current eventspace, as determined by \texttt{current-eventspace}. Otherwise, \texttt{parent}'s eventspace is the new frame’s eventspace.

If the \texttt{width} or \texttt{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 \texttt{x} or \texttt{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 \texttt{style} flags adjust the appearance of the frame on some platforms:

- `no-resize-border` — omits the resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- `no-caption` — omits the title bar for the frame (Windows)
- `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)

If the `mdi-child` style is specified, the \texttt{parent} must be a frame with the `mdi-parent` style, otherwise an \texttt{exn:application:mismatch} 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.

11.29 \texttt{frame:searchable\%} = (frame:searchable-mixin frame:text-info-file\%)

\texttt{frame:searchable\%} = (frame:searchable-mixin frame:text-info-file\%)

- (make-object frame:searchable\% label parent width height x y style) \Rightarrow \texttt{frame:searchable\%}

  \texttt{object}

  \texttt{label : string}

  \texttt{parent = \#f : frame\% object or \#f}
11.30. \texttt{frame:pasteboard\%} = (frame:pasteboard-mixin frame:editor\%)

\texttt{frame:pasteboard\%} = (frame:pasteboard-mixin frame:editor\%)

- (make-object frame:pasteboard\% \texttt{label} parent width height x y style) ⇒ frame:pasteboard\%

  \texttt{label} : string
  \texttt{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 \([0, 10000]\) or \#f
  y = \#f : exact integer in \([0, 10000]\) or \#f
  \texttt{style} = null : list of symbols in ’(no-resize-border no-caption no-system-menu mdi-parent mdi-child)

The \texttt{label} string is displayed in the frame’s title bar. If the frame’s label is changed (see \texttt{set-label} ), the title bar is updated.

The \texttt{parent} argument can be \#f or an existing frame. Under Windows, if \texttt{parent} is an existing frame, the new frame is always on top of its parent. Also, the \texttt{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 \texttt{parent} is \#f, then the eventspace for the new frame is the current eventspace, as determined by \texttt{current-eventspace}. Otherwise, \texttt{parent}’s eventspace is the new frame’s eventspace.

If the \texttt{width} or \texttt{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 \texttt{x} or \texttt{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 \texttt{style} flags adjust the appearance of the frame on some platforms:

- ’no-resize-border — omits the resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- ’no-caption — omits the title bar for the frame (Windows)
- ’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)

If the ’mdi-child style is specified, the \texttt{parent} must be a frame with the ’mdi-parent style, otherwise an \texttt{exn:application:mismatch} 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.
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 resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- `'no-caption` — omits the title bar for the frame (Windows)
- `'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)

If the `'mdi-child` style is specified, the `parent` must be a frame with the `'mdi-parent` style, otherwise an `exn:application:mismatch` 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.

### 11.31 frame:pasteboard-info-file% = (frame:file-mixin frame:pasteboard%)
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 \textit{style} flags adjust the appearance of the frame on some platforms:

- \texttt{'no-resize-border} — omits the resizeable border around the window (Windows) or grow box in the bottom right corner (MacOS)
- \texttt{'no-caption} — omits the title bar for the frame (Windows)
- \texttt{'no-system-menu} — omits the system menu (Windows)
- \texttt{'mdi-child} — creates the frame as a MDI (multiple document interface) child frame, mutually exclusive with \texttt{'mdi-parent} (Windows)
- \texttt{'mdi-parent} — creates the frame as a MDI (multiple document interface) parent frame, mutually exclusive with \texttt{'mdi-child} (Windows)

If the \texttt{'mdi-child} style is specified, the \textit{parent} must be a frame with the \texttt{'mdi-parent} style, otherwise an \texttt{exn:application:mismatch} 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.

\section*{11.32 Frame Utilities}

\texttt{frame:reorder-menus}

Re-orders the menus in a frame. This is useful in conjunction with the \texttt{frame:standard-menus} class. After instantiating that class and adding menus, the menus will be mis-ordered. This will put the File and Edit menus at the front of the menubar and the Help menu at the end.

\begin{verbatim}
- (frame:reorder-menus frame) ⇒ void
  frame : (instance frame%)
\end{verbatim}
12. 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.

12.1 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.

can-remove-frame?

- (send a-group: can-remove-frame?) ⇒ boolean

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.

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<%>))

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-empty-callbacks**

Sets the empty callbacks. These functions are called when the frame group is empty.

- (send a-group: set-empty-callbacks test close-down) ⇒ void

  _test_ : (-¿ boolean)
  
  _close-down_ : (-¿ void)

  The function _test_ is called when there is one frame left in the group. If it returns #t, the closing operation may be completed. If it returns #f, the closing operation is aborted.

  The function _close-down_ is called when there are no frames left in the group.

### 12.2 Group Utilities

**group:get-the-frame-group**

This returns the frame group.

- (group:get-the-frame-group) ⇒ (instance group:<%>)
13. GUI Utilities

This section has several utility for GUI programs.

13.1 gui-utils:text-snip

Objects that implement this interface are treated specially by gui-utils:read-snips/chars-from-text.

get-string

See gui-utils:read-snips/chars-from-text for more information.

- (send a-gui-utils:text-snip get-string) ⇒ string

13.2 Gui-utils Utilities

gui-utils:cursor-delay

This function is not a parameter.

- (gui-utils:cursor-delay) ⇒ real
  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.

- (gui-utils:cursor-delay new-delay) ⇒ void
  new-delay : real
  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

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
        (lambda () .. init watch cursor ...)
        (lambda () .. close watch cursor ...))])

  .. do action ...
  (close-down))
- (gui-utils:delay-action delay-time open close) ⇒ (-¿ void)
  delay-time : real
  open : (-¿ void)
  close : (-¿ void)

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

Opens a dialog that presents a binary choice to the user. The user is forced to choose between these two options, i.e. cancelling or closing the dialog opens a message box asking the user to actually choose one of the two options.

- (gui-utils:get-choice message true-choice false-choice title default-result parent) ⇒ boolean
  message : string
  true-choice : string
  false-choice : string
  title = "Warning" : string
  default-result = 'disallow-close : (union 'disallow-close TST)
  parent : (union frame% dialog% #f)

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.

gui-utils:get-clickback-delta

- (gui-utils:get-clickback-delta) ⇒ (instance style-delta%)
  this delta is designed for use with set-clickback. Set the text that is clickable to this style-delta. It turns the text blue and underlines it.

gui-utils:get-clicked-clickback-delta

- (gui-utils:get-clicked-clickback-delta) ⇒ (instance style-delta%)
  this delta is designed for use with set-clickback. Use it as the style-delta% argument to set-clickback.

gui-utils:local-busy-cursor

- (gui-utils:local-busy-cursor window thunk delay) ⇒ A
  window : (union #f (implements window<%>))
  thunk : (-¿ A)
  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

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:next-untitled-name) ⇒ string

gui-utils:open-input-buffer

- (gui-utils:open-input-buffer text) ⇒ input-port
  text : text% object
  This procedure returns a port that reads characters from the buffer.

gui-utils:read-snips/chars-from-text

This function returns a thunk, which when called returns all of the characters and non-text-snips from the edit, one by one.

If this function encounters a snip that matches gui-utils:text-snip<%>, it uses the get-string method to get a string from the snip and returns the characters in that.

- (gui-utils:read-snips/chars-from-text text start end) ⇒ (-¿ (union char (instance snip% e o f - object))
  text : (instance text%)
  start = 0 : integer
  end = (send text last-position) : integer

gui-utils:show-busy-cursor

- (gui-utils:show-busy-cursor thunk delay) ⇒ A
  thunk : (-¿ A)
  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:unsaved-warning

This displays a dialog that warns the user of a unsaved file.

- (gui-utils:unsaved-warning filename action can-save-now? parent) ⇒ (union 'continue 'save 'cancel)
  filename : string
  action : string
  can-save-now? = #f : boolean
  parent = #f : (union #f (instance frame%) (instance dialog%))
  The string, action, indicates what action is about to take place. For example, if the application is about to close a file, a good action is "Close". 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.
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-info-file%` 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`.

14.1 Handler Utilities

`handler:edit-file`

This function creates a frame to edit a file.

It invokes the appropriate format handler to open the file (see `handler:insert-format-handler`).

```
- (handler:edit-file filename make-default) ⇒ (implements frame:editor<%>)
  filename : (union string #f)
  make-default = (lambda () (make-object frame:text-info-file% filename))
  - 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`

This function selects a format handler. See also `handler:insert-format-handler`.

```
- (handler:find-format-handler filename) ⇒ (string -¿ (implements frame:editor<%>))
  filename : string
  It finds a handler based on filename.
```

`handler:find-named-format-handler`

This function selects a format handler. See also `handler:insert-format-handler`. 
- (handler:find-named-format-handler name) ⇒ (string -¥ (implements frame:editor<%>))
  name : string

  It finds a handler based on name.

handler:insert-format-handler

This function inserts a format handler.

- (handler:insert-format-handler name pred handler) ⇒ void
  name : string
  pred : (union string (listof string) (string -¥ boolean))
  handler : (string -¥ (implements frame:editor<%>))

  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.

handler:open-file

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.

- (handler:open-file) ⇒ (instance frame:basic<%>)

15. Icon

15.1 Icon Utilities

icon:get-anchor-bitmap
This returns the anchor’s bitmap%.

- (icon:get-anchor-bitmap) ⇒ (implements bitmap%)

icon:get-autowrap-bitmap
This returns the autowrap’s bitmap%.

- (icon:get-autowrap-bitmap) ⇒ (implements bitmap%)

icon:get-gc-off-bitmap

- (icon:get-gc-off-bitmap) ⇒ (instance bitmap%)
  This returns a bitmap to be displayed in an frame:info<%> frame when garbage collection is not taking place.

icon:get-gc-on-bitmap

- (icon:get-gc-on-bitmap) ⇒ (instance bitmap%)
  This returns a bitmap to be displayed in an frame:info<%> frame when garbage collection is taking place.

icon:get-lock-bitmap
This returns the lock’s bitmap%.

- (icon:get-lock-bitmap) ⇒ (implements bitmap%)

icon:get-paren-highlight-bitmap
This returns the parenthesis highlight bitmap%. It is only used on black and white screens.

- (icon:get-paren-highlight-bitmap) ⇒ (implements bitmap%)
icon:get-unlock-bitmap

This returns the reset unlocked bitmap.

- (icon:get-unlock-bitmap) ⇒ (implements bitmap)
16. Keymap

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%

- (make-object keymap:aug-keymap-mixin%) ⇒ keymap:aug-keymap-mixin% object

Creates an empty keymap.
16.2. keymap: aug-keymap-mixin

keymap

Multiple keymaps can be chained off one keymap using `chain-to-keymap`. When keymaps are chained to a main keymap, then events handled by the main keymap are passed to the chained keymaps until a 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 by chained groups. Dispatching each individual event to separate keymaps is problematic without chaining because keymaps may acquire state that must be reset when a callback is invoked in one of the keymaps. This state can be manually cleared with `break-sequence`, but this also invokes the handler installed by `set-break-sequence-callback`.

```lisp
- (send a-keymap:aug-keymap-mixin chain-to-keymap next prefix?) ⇒ void
  next : (instance keymap%)
  prefix? : boolean
```

If `next` will be used to handle events which are not handled by this keymap. If `prefix?` is a true value, then `next` will take precedence over other keymaps already chained to this one.

Keeps a list of the keymaps chained to this one.

map-function

Maps an input state to the name of an event handler.

```lisp
- (send a-keymap:aug-keymap-mixin map-function key-name function-name) ⇒ void
  key-name : string
  function-name : string
```

Maps an input state sequence to a function name using a string-encoded sequence in `keyname`. The format of `keyname` is a sequence of semicolon-delimited input states; each state is made up of a sequence of modifier identifiers followed by a key identifier.

The modifier identifiers are:
- “s:” — All platforms: Shift
- “c:” — All platforms: Control
- “a:” — MacOS: Option
- “m:” — Windows: Alt; X: Meta
- “d:” — MacOS: Command

If a particular modifier is not mentioned in a state string, it matches states whether that modifier is pressed or not pressed. A tilde ( ) preceding a modifier makes the string match only states where the corresponding modifier is not pressed. If the state string begins with a colon, then the string only matches a state if modifiers not mentioned in the string are not pressed.

A key identifier can be either a character on the keyboard (e.g., "a", "2", "?”) or a special name. The special names are:
- "leftbutton” (button down)
- "rightbutton"
- "middlebutton"
- "leftbuttondouble" (button down for double-click)
- "rightbuttondouble"
- "middlebuttondouble"
- "leftbuttontriple" (button down for triple-click)
- "rightbuttontriple"
- "middlebuttontriple"
- "leftbuttonseq" (all events from button down through button up)
- "rightbuttonseq"
- "middlebuttonseq"
- "esc"
- "delete"
- "del" (same as "delete")
- "insert"
- "ins" (same as "insert")
- "add"
- "subtract"
- "multiply"
- "divide"
- "backspace"
- "back"
- "return"
- "enter" (same as "return")
- "tab"
- "space"
- "right"
- "left"
- "up"
- "down"
- "home"
- "end"
- "pageup"
- "pagedown"
- "semicolon"
- "colon"
- "numpad1"
- "numpad2"
- "numpad3"
- "numpad4"
- "numpad5"
- "numpad6"
- "numpad7"
- "numpad8"
- "numpad9"
- "f1"
- "f2"
- "f3"
- "f4"
- "f5"
- "f6"
- "f7"
- "f8"
- "f9"
- "f10"
- "f11"
- "f12"
- "f13"
- "f14"
- "f15"
- "f16"
- "f17"
For a special keyword, the capitalization does not matter. However, capitalization is important for single-letter keynames (e.g., "A" is interpreted as "s:a").

A state can match multiple state strings mapped in a keymap (or keymap chain); when a state matches multiple state strings, a mapping is selected by ranking the strings according to specificity. A state string that mentions more pressed modifiers has a higher rank than other state strings, and if two strings mention the same number of pressed modifiers, the one that mentions more unpressed modifiers has a higher rank. In that case that multiple matching strings have the same rank, one string is selected arbitrarily.

Examples:
- "space" — matches whenever the space bar is pressed, regardless of the state of modifiers keys.
- "c:space" — matches whenever the space bar is pressed and the Control key is not pressed.
- "a" — matches whenever “a” is typed, regardless of the state of modifiers keys other than Shift.
- "a:" — matches only when “a” is typed with no modifier keys pressed.
- "c:a" — matches whenever “a” is typed and neither the Shift key nor the Control key is pressed.
- ":esc;c:c" — matches an Escape key press (no modifiers) followed by a Control-C press (no modifiers other than Control).

A call to \texttt{map-function} that would map a particular key sequence both as a prefix and as a complete sequence raises an exception, but the exception handler cannot escape (see Exceptions and Continuation Jumps, §2.4.4 in \textit{PLT MrEd: Graphical Toolbox Manual}).

A function name does not have to be mapped to a handler before input states are mapped to the name; the handler is dispatched by name at the time of invocation. The event handler mapped to a function name can be changed without affecting the map from input states to function names.

Keeps a separate record of the key names and functions that they are bound to in this keymap.

16.3 \texttt{keymap:aug-keymap\% = (keymap:aug-keymap-mixin keymap\%)}

\texttt{keymap:aug-keymap\% = (keymap:aug-keymap-mixin keymap\%)}

- (make-object keymap:aug-keymap\%) ⇒ keymap:aug-keymap\% object
  Creates an empty keymap.

16.4 \texttt{Keymap Utilities}

\texttt{keymap:call/text-keymap-initializer}

- (keymap:call/text-keymap-initializer thunk-proc) ⇒ A
  \hspace{1em} \texttt{thunk-proc : (→ A)}
  Parameterize the call to \texttt{thunk-proc} by setting the keymap-initialization procedure (see \texttt{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

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-editor) ⇒ (instance keymap%)

keymap:get-file

This returns a keymap for handling file operations.

- (keymap:get-file) ⇒ (instance keymap%)

keymap:get-global

This returns a keymap for general operations. See keymap:setup-global for a list of the bindings this keymap contains.

- (keymap:get-global) ⇒ (implements keymap%)

keymap:get-search

This returns a keymap for searching operations

- (keymap:get-search) ⇒ (implements keymap%)
16.4. Keymap Utilities

keymap:make-meta-prefix-list

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:make-meta-prefix-list key) ⇒ (listof string)
  key : string

keymap:send-map-function-meta

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.

- (keymap:send-map-function-meta keymap key func) ⇒ void
  keymap : (instance keymap%)
  key : string
  func : (TST (instance key-event%) -¿ boolean)

This procedure binds all of the key-bindings obtained by prefixing key with a meta-prefix to func in keymap.

keymap:setup-editor

This sets up the input keymap with the bindings described in keymap:get-editor.

- (keymap:setup-editor keymap) ⇒ void
  keymap : (instance keymap%)

keymap:setup-file

This extends a keymap% with the bindings for files.

- (keymap:setup-file keymap) ⇒ void
  keymap : (instance keymap%)

keymap:setup-global

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 as “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-I : “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”
- (keymap:setup-global keymap) ⇒ void
  keymap : (instance keymap%)

keymap:setup-search

This extends a keymap% with the bindings for searching.

- (keymap:setup-search keymap) ⇒ void
  keymap : (instance keymap%)
17. Keys

17.1 Keys Utilities

**keys:get-shifted-key-list**

- (keys:get-shifted-key-list) ⇒ (listof string)
  
  This returns a list of the keys that are typed using shift.
  
  It returns:

  ```
  (list "?" ":" ":" ":" ":" ":" ":"
   ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":"
   ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":"
   ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":" ":"
  )
  ```

  This is not specific to a particular keyboard, but it should be.
18. Main
19. Match Cache

19.1 match-cache:%

This class defines a cache that can be used with paren:backward-match and paren:forward-match. The cache data is intended to be inserted and read only by the matching procedures; however, the cache must be specifically invalidated when the cache’s buffer is modified.

A cache is not required by the matching procedures. A single cache can only be used for a single buffer and with a single direction and parenthesis, quote, and comment parameterization. When a cache is used for a buffer, it does not have to be used with every call to the matching procedures, but the best results are obtained when the cache is always used. Multiple caches can be used for a single buffer (possibly for different directions or parenthesis parameterizations), as long as they are all invalidated properly when the buffer is modified.

A text’s after-insert and after-delete methods can be overridden to properly track modifications and invoke a cache’s invalidate or forward-invalidate methods.

contents
UNDOCUMENTED

- (send a-match-cache: contents) ⇒ void

delete
UNDOCUMENTED

- (send a-match-cache: delete) ⇒ void

forward-invalidate

- (send a-match-cache: forward-invalidate pos) ⇒ void
  pos : integer

  Call this method when text is inserted or deleted at position pos in the cache’s buffer and the cache is used for forward-matching.

get
UNDOCUMENTED

- (send a-match-cache: get) ⇒ void
invalidate

- (send a-match-cache: invalidate pos) ⇒ void
  pos : integer
  Call this method when text is inserted or deleted at position pos in the cache’s buffer and the cache is used for backward-matching.

max-count

UNDOCUMENTED

- (send a-match-cache: max-count) ⇒ void

put

UNDOCUMENTED

- (send a-match-cache: put) ⇒ void

splay

UNDOCUMENTED

- (send a-match-cache: splay) ⇒ void

sum

UNDOCUMENTED

- (send a-match-cache: sum) ⇒ void

times

UNDOCUMENTED

- (send a-match-cache: times) ⇒ void
20. Panel

area-container<%

| panel:single<%> window<%>
|------------------------|
+------------------------+
| panel:single-window<%>

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<%>

  Does nothing.
  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: window<%>

Extends: panel:single<%>

20.4 panel:single-window-mixin

Domain: window<%>

Domain: panel:single<%>

Implements: panel:single-window<%>
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-window-mixin (panel:single-mixin panel%))

panel:single% = (panel:single-window-mixin (panel:single-mixin panel%))

- (make-object panel:single% parent style) ⇒ panel:single% object
  parent : frame%, dialog%, panel%, or pane% object
  style = null : list of symbols in 'border'
  If the 'border style is specified, the window is created with a thin border (only in this case, the client size of the panel may be less than its total size).

20.6 panel:single-pane% = (panel:single-mixin pane%)

panel:single-pane% = (panel:single-mixin pane%)

- (make-object panel:single-pane% parent) ⇒ panel:single-pane% object
  parent : frame%, dialog%, panel%, or pane% object
21. Parenthesis

MrEd provides general-purpose “parenthesis”-matching utilities that work on buffers. The utilities are parameterized with respect to:

- **parens** — Pairs (cons cells) of opening and closing bracket strings. These brackets can be nested and must be balanced. The opening and closing string do not have to be different.

- **quotes** — Pairs of opening and closing quote bracket strings. Within a pair of quotes, all other bracket, quote, and comment strings are ignored. Pairs of quote strings do not have to correspond to quotes in the language. For example, C comments using “/*” and “*/” are considered quotes for parenthesis-matching purposes.

- **comments** — A list of comment-starting strings. Comments of this form run until the end of the line.

A backslash (\) is assumed to be the (only) method for escaping parenthesis, quote, and comment markers. The parenthesis-balancing utilities are only guaranteed to act meaningfully when the starting position is outside quoted and commented expressions. When a pair of opening and closing brackets are not distinct, the actions on these brackets are only meaningful for searches starting outside the brackets.

### 21.1 Paren Utilities

**paren:backward-match**

- (paren:backward-match text start end parens quotes comments containing? cache) ⇒ (union integer #f)
  
  text : (instance text%)
  start : exact-integer
  end : exact-integer
  parens : (listof (cons string string))
  quotes : (listof (cons string string))
  comments : (listof string)
  containing? = #f : boolean
  cache = #f : (union #f (instance match-cache:%))

Returns the position in text that “opens” the text ending at start, or #f if no opening position is found (either because a parenthesis mis-match is discovered or the end boundary was reached). The match must occur before end (inclusive). Note that start > end, since start specifies the starting position of the search, not the earliest buffer position to be considered.

Spaces immediately preceding start are skipped. If the text at start is a close parenthesis or close quote, then the matching position is the opening parenthesis or quote. If a comment immediately precedes start, then the comment is skipped as whitespace. If an opening parenthesis immediately precedes start, then the matching position is start - 1. Otherwise, the matching position is the first whitespace or parenthesis character before start.

If containing? is not #f, then the matching procedure is modified as follows:
– Searching iterates backwards until some search fails. Then, the location of the last successful search is returned.
– If a mis-match is detected, then #f is returned.
– If there are no matches (and no mis-matches) before start, start itself is returned.

If cache is not #f, it must be an instance of match-cache:%. A cache object can be used to speed up successive calls to paren:backward-match. However, a buffer using a cache must call the cache’s invalidate method when the buffer is modified. Different caches should be used for forward and backward matching. See section 19.1 for more information.

paren:balanced?

- (paren:balanced? text start end parens quotes comments) ⇒ boolean
  text : (instance text%)
  start : exact-integer
  end : exact-integer
  parens : (listof (cons string string))
  quotes : (listof (cons string string))
  comments : (listof string)

Returns #t if the text in text between positions start and end is balanced. The text is balanced if there are no unclosed parentheses or quotes, there are no closing parentheses that do not match an open parenthesis, and there are no mis-matched parentheses.

This uses paren:forward-match.

paren:forward-match

- (paren:forward-match text start end parens quotes comments cache) ⇒ (union #f integer)
  text : (instance text%)
  start : exact-integer
  end : exact-integer
  parens : (listof (cons string string))
  quotes : (listof (cons string string))
  comments : (listof string)
  cache : (union #f (instance match-cache:%))

This function returns the position in text that “closes” the text at start, or #f if no closing position is found (either because a parenthesis mis-match is discovered or the end boundary was reached). The match must occur before end (inclusive).

Spaces immediately following start are skipped. If the text at start is an open parenthesis or open quote, then the matching position is the closing parenthesis or quote. If a comment immediately follows start, it is skipped over as whitespace. If a closing parenthesis immediately follows start (after skipping whitespace), then #f is returned. Otherwise, the matching position is the position before the first whitespace, parenthesis, quote, or comment character after start.

If cache is not #f, it must be an instance of match-cache:%. A cache object can be used to speed up successive calls to paren:forwardward-match. However, a buffer using a cache must call the cache’s forward-invalidate method when the buffer is modified. Different caches should be used for forward and backward matching. See section 19.1 for more information.

paren:skip-whitespace

- (paren:skip-whitespace text pos dir) ⇒ exact-integer
  text : (instance text%)
pos : exact-integer

dir : (union 'forward 'backward)

If dir is 'forward, this returns the position of the first non-whitespace character in text after pos. If dir is 'backward, it returns the first non-whitespace character before pos.
22. Pasteboard

22.1 pasteboard:basic% = (editor:basic-mixin pasteboard%)

pasteboard:basic% = (editor:basic-mixin pasteboard%)

- (make-object pasteboard:basic%) ⇒ pasteboard:basic% object
  The editor will not be displayed until it is attached to a editor-canvas% object or some other display.
  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.

22.2 pasteboard:keymap% = (editor:keymap-mixin pasteboard:basic%)

pasteboard:keymap% = (editor:keymap-mixin pasteboard:basic%)

- (make-object pasteboard:keymap%) ⇒ pasteboard:keymap% object
  The editor will not be displayed until it is attached to a editor-canvas% object or some other display.
  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.

22.3 pasteboard:file% = (editor:file-mixin pasteboard:keymap%)

pasteboard:file% = (editor:file-mixin pasteboard:keymap%)

- (make-object pasteboard:file%) ⇒ pasteboard:file% object
  The editor will not be displayed until it is attached to a editor-canvas% object or some other display.
  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.

22.4 pasteboard:backup-autosave% = (editor:backup-autosave-mixin pasteboard:file%)

pasteboard:backup-autosave% = (editor:backup-autosave-mixin pasteboard:file%)

- (make-object pasteboard:backup-autosave%) ⇒ pasteboard:backup-autosave% object
  The editor will not be displayed until it is attached to a editor-canvas% object or some other display.
  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.
22.5 pasteboard:info% = (editor:info-mixin pasteboard:backup-autosave%)
23. Pathname Utilities

23.1 Path-utils Utilities

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) ⇒ string
  
  filename : string

  Generates a name for a backup file from filename.
24. 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.

In addition to the functions @flink preferences:add-general-panel, and @flink preferences:add-font-panel listed here, @flink scheme:add-preferences-panel also adds panels to the preferences dialog.

24.1 Preferences Utilities

preferences:add-callback

- (preferences:add-callback p f) ⇒ (-没事 void)
  
  p : symbol
  f : (symbol TST -没事 boolean)

  This function adds a callback which is called with a symbol naming a preference and its value, when the preference changes. If the callback function returns #f, the preference is not changed. preferences:add-callback returns a thunk, which when invoked, removes the callback from this preference.

  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-font-panel

Adds a font selection preferences panel to the preferences dialog.

- (preferences:add-font-panel) ⇒ void

preferences:add-general-panel

Adds a general preferences panel to the preferences dialog.

- (preferences:add-general-panel) ⇒ void

preferences:add-panel

- (preferences:add-panel name f) ⇒ void
  
  name : string
  f : ((instance area-container%) -没事 (instance area-container%))
preferences:add-preference-panel adds the result of \( f \) with name \( name \) to the preferences dialog box. \( f \)'s result's parent must be \( f \)'s argument.

preferences:get
See also preferences:set-default.

\[-(\text{preferences:}\!\gt\text{get } symbol) \Rightarrow \text{TST}\]
\[symbol : \text{symbol}\]
preferences:get returns the value for the preference \( symbol \). It raises \text{exn:unknown-preference} if the preference has not been set.

preferences:hide-dialog
This function hides the preferences dialog.

\[-(\text{preferences:}\!\gt\text{hide-dialog}) \Rightarrow \text{void}\]

preferences:read
Reads the preferences from the preferences file and installs their values.

\[-(\text{preferences:}\!\gt\text{read}) \Rightarrow \text{void}\]

preferences:restore-defaults

\[-(\text{preferences:}\!\gt\text{restore-defaults}) \Rightarrow \text{void}\]
\[\text{(preferences:restore-defaults)} \text{ restores the users's configuration to the default preferences.}\]

preferences:save

\[-(\text{preferences:}\!\gt\text{save}) \Rightarrow \text{void}\]
\[\text{(preferences:save-user-preferences)} \text{ saves the user's preferences to disk, potentially marshalling some of the preferences.}\]

preferences:set

\[-(\text{preferences:}\!\gt\text{set } symbol \text{ value}) \Rightarrow \text{void}\]
\[symbol : \text{symbol}\]
\[value : \text{TST}\]
preferences:set-preference sets the preference \( symbol \) to \( value \). This should be called when the users requests a change to a preference.

preferences:set-default
This function must be called every time your application starts up, before any call to preferences:get.

If you use preferences:set-un/marshall, you must also call it before calling this function.
24. Preferences

24.1. Preferences Utilities

- (preferences:set-default symbol value test) ⇒ void
  symbol : symbol
  value : TST
  test : (TST -¿ boolean)

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.

If there is a site-wide default preferences file, the default preference in that file is used instead of value.

preferences:set-un/marshall

- (preferences:set-un/marshall symbol marshall unmarshall) ⇒ void
  symbol : symbol
  marshall : (TST -¿ printable)
  unmarshall : (printable -¿ TST)

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 it’s internal representation. If preferences:set-un/marshall is never called for a particular preference, the values of that preference are assumed to be printable.

preferences:set-un/marshall must be called before calling preferences:get% or preferences:set-default%.

preferences:show-dialog

This function shows the preferences dialog.

- (preferences:show-dialog) ⇒ void
25. Scheme

25.1 scheme: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%)

balance-quotes

This function is called when the user types a quote in the text%. If the framework:paren-match preference is #t (see preferences:get) the matching open quote is flashed.

- (send a-scheme:text balance-quotes key-event) ⇒ void
  
  key-event : (instance key-event%)

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
25.1. scheme:text<%>

Returns the position of the end of next S-expression after position \textit{start}, or \texttt{#f} if there is no appropriate answer.

\textbf{get-limit}

- \texttt{(send a-scheme:text get-limit start)} \Rightarrow \texttt{int}
  
  \texttt{start : exact-integer}

  Returns a limit for backward-matching parenthesis starting at position \texttt{start}.

\textbf{get-tab-size}

This method returns the current size of the tabs for scheme mode. See also \texttt{set-tab-size}.

- \texttt{(send a-scheme:text get-tab-size)} \Rightarrow \texttt{exact-integer}

\textbf{highlight-parens}

- \texttt{(send a-scheme:text highlight-parens just-clear?)} \Rightarrow \texttt{void}
  
  \texttt{just-clear? = #f : boolean}

  When the current position is at the beginning or the end of a sexpression, this method highlights the sexpression.

\textbf{insert-return}

- \texttt{(send a-scheme:text insert-return)} \Rightarrow \texttt{void}

  Inserts a newline into the buffer. If \texttt{tabify-on-return?} returns \texttt{#t}, this will tabify the new line.

\textbf{remove-parens-forward}

- \texttt{(send a-scheme:text remove-parens-forward start)} \Rightarrow \texttt{void}
  
  \texttt{start : exact-integer}

  Removes the parentheses from the S-expression starting after the position \texttt{start}.

\textbf{remove-sexp}

- \texttt{(send a-scheme:text remove-sexp start)} \Rightarrow \texttt{void}
  
  \texttt{start : exact-integer}

  Forward-deletes the S-expression starting after the position \texttt{start}.

\textbf{select-backward-sexp}

- \texttt{(send a-scheme:text select-backward-sexp start)} \Rightarrow \texttt{#t}
  
  \texttt{start : exact-integer}

  Selects the previous S-expression, starting at position \texttt{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 \textit{start}.

select-forward-sexp

- (send a-scheme:text select-forward-sexp start) ⇒ #t
  start : exact-integer
  Selects the next S-expression, starting at position \textit{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 \textit{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 \textit{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
25.2 scheme:text-mixin

Domain: text:basic<%>

Implements: scheme: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.

after-change-style

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-scheme:text-mixin after-change-style start len) ⇒ void
  
  start : exact non-negative integer
  len : exact non-negative integer
This calls \texttt{end-edit-sequence} and updates the parenthesis highlight. See \texttt{highlight-parens}.

\textbf{after-delete}

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

See also \texttt{can-delete?} and \texttt{on-edit-sequence}.

No internals locks are set when this method is called.

\begin{verbatim}
- (send a-scheme:text-mixin after-delete start end) ⇒ void
  start : exact non-negative integer
  end : exact non-negative integer
  The \textit{start} argument specifies the starting position of the deleted range. The \textit{len} argument specifies number of deleted items (so \textit{start} + \textit{length} is the endig position of the deleted range).
  This calls \texttt{end-edit-sequence} and updates the parenthesis highlight. See \texttt{highlight-parens}.
\end{verbatim}

\textbf{after-edit-sequence}

Called after a top-level edit sequence completes (involving unnested \texttt{begin-edit-sequence} and \texttt{end-edit-sequence}).

See also \texttt{on-edit-sequence}.

\begin{verbatim}
- (send a-scheme:text-mixin after-edit-sequence) ⇒ void
  This updates the parenthesis highlight. See \texttt{highlight-parens}.
\end{verbatim}

\textbf{after-insert}

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

See also \texttt{can-insert?} and \texttt{on-edit-sequence}.

No internals locks are set when this method is called.

\begin{verbatim}
- (send a-scheme:text-mixin after-insert start len) ⇒ void
  start : exact non-negative integer
  len : exact non-negative integer
  The \textit{start} argument specifies the position of the insert. The \textit{len} argument specifies the total length (in positions) of the inserted items.
  This calls \texttt{end-edit-sequence} and updates the parenthesis highlight. See \texttt{highlight-parens}.
\end{verbatim}

\textbf{after-set-position}

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

See also \texttt{on-edit-sequence}.
25.3 scheme:text% = (scheme:text-mixin text:info%)

- (send a-scheme:text-mixin after-set-position) ⇒ void
  This calls end-edit-sequence and updates the parenthesis highlight. See highlight-parens.

after-set-size-constraint

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-scheme:text-mixin after-set-size-constraint) ⇒ void
  This calls end-edit-sequence and updates the parenthesis highlight. See highlight-parens.

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-scheme:text-mixin on-focus on?) ⇒ boolean
  on? : boolean
  This calls begin-edit-sequence.

25.3 scheme:text% = (scheme:text-mixin text:info%)

scheme:text% = (scheme:text-mixin text:info%)

- (make-object scheme:text% line-spacing tabstops) ⇒ scheme:text% object
  line-spacing = 1.0 : non-negative real number
  tabstops = null : list of real numbers

  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.

  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.

25.4 Scheme Utilities

scheme:add-preferences-panel

Adds a tabbing preferences panel to the preferences dialog.

- (scheme:add-preferences-panel) ⇒ void
scheme:get-keymap
Returns a keymap with binding suitable for Scheme.

- (scheme:get-keymap) ⇒ (instance keymap%)

scheme:get-style-list
Returns a style list that is used for all Scheme buffers.

- (scheme:get-style-list) ⇒ (instance style-list%)

scheme:get-wordbreak-map
This method returns an editor-wordbreak-map% that is suitable for Scheme.

- (scheme:get-wordbreak-map) ⇒ (instance editor-wordbreak-map%)

scheme:init-wordbreak-map

- (scheme:init-wordbreak-map key) ⇒ void
  key : (instance keymap%)
  Initializes the workdbreak map for keymap.

scheme:setup-keymap

- (scheme:setup-keymap keymap) ⇒ void
  keymap : (instance keymap%)
26. Scheme Parenthesis

26.1 Scheme-paren Utilities

scheme-paren:backward-containing-sexp

  - (scheme-paren:backward-containing-sexp text start end cache) ⇒ exact-integer
    text : (instance text%)  
    start : exact-integer  
    end : exact-integer  
    cache : (union #f (instance match-cache:%))

  Returns the beginning of the interior of the (non-atomic) S-expression containing start.

scheme-paren:backward-match

Specializes paren:backward-match to Scheme.

  - (scheme-paren:backward-match text start end cache) ⇒ exact-integer
    text : (instance text%)  
    start : exact-integer  
    end : exact-integer  
    cache = #f : (union #f (instance match-cache:%))

scheme-paren:balanced?

Specializes paren:balanced? to Scheme.

  - (scheme-paren:balanced? text start end) ⇒ boolean
    text : (instance text%)  
    start : exact-integer  
    end : exact-integer

scheme-paren:forward-match

Specializes paren:forward-match to Scheme.

  - (scheme-paren:forward-match text start end cache) ⇒ (union #f integer)
    text : (instance text%)  
    start : exact-integer  
    end : exact-integer  
    cache : (union #f (instance match-cache:%))
scheme-paren:get-comments

Returns the comment characters for Scheme.

- (scheme-paren:get-comments) ⇒ (listof string)

scheme-paren:get-paren-pairs

Returns the paren pairs for Scheme.

- (scheme-paren:get-paren-pairs) ⇒ (listof (cons string string))

scheme-paren:get-quote-pairs

Returns the quote pairs for Scheme.

- (scheme-paren:get-quote-pairs) ⇒ (listof (cons string string))
27. Test

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 section ?? 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.

27.1 An example

Here is an example program that enters a factorial procedure and computes \((\text{fact } 4)\). To run this program, start DrScheme, click on the “Console” button, load this program and run \((\text{go})\). Then bring the DrScheme window to the front and click the mouse in the DrScheme window.

```
(define go
  (lambda ()
    (sleep 3)
    (test:new-window (get-panel '(0 0 0 1))) ; definitions canvas
    (test:menu-select "Edit" "Select All")
    (test:menu-select "Edit" "Delete")
    (type-line "(define fact")
    (type-line "((lambda n)"
    (type-line "((if (zero? n)"
    (type-line "1")
    (type-line "(* n (fact (sub1 n))))))")
    (test:button-push (get-panel '(0 0 0 0 5 0))) ; check-syntax button
    (test:button-push (get-panel '(0 0 0 0 5 3))) ; execute button
    (sleep 3)
    (type-line "(fact 4")
    (sleep 1)
    (printf "Test complete. Pending actions: "
      (test:number-pending-actions)))
```

132
27.2 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.

27.3 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.
27.4 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 see section 2.4 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
```

27.5 Known Problems

The following are known problems or not yet implemented.

1. Support for listboxes, sliders and radioboxes has not been implemented.
2. Support for mouse dragging and double clicking has not been implemented.
3. Using `test:menu-select` for check-boxes on menu bars (for example, “Show” in DrScheme) does not work.
4. The parameterization for the DrScheme console REPL is shared with the handler thread. As a result, some errors in the main thread are not handled correctly. The simplest solution is to reset the error handler in the test suite with:

   ```lisp
   (current-error-handler (current-error-handler))
   ```

5. Sending keystrokes to a `mred:text%` window converts everything to upper case.

27.6 Test Utilities

`test:button-push`

- `(test:button-push button) ⇒ void`
  
  *button* : (union (instance button%) string)

  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`

Use this function to simulate clicking on the close box of a frame.
27. Test

27.6. Test Utilities

- (test:close-top-level-window frame) ⇒ void
  
  Closes frame with this expression:

  (when (send frame can-close?)
   (send frame on-close)
   (send frame show #f))

**test:current-get-eventspaces**

This parameter that specifies which eventspace (see section 2.4) are considered when finding the frontmost frame.

- (test:current-get-eventspaces func) ⇒ void
  
  func : (-¿ (listof eventspace))

  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.

- (test:current-get-eventspaces) ⇒ (-¿ (listof eventspace))

  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 (union 'alt 'control 'meta 'shift 'noalt 'nocontrol 'nometa '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 \#return, not \#newline.

  The ‘shift or ‘noshift modifier is implicitly set from key, but is overridden by the argument list. The ‘shift modifier is set for any capital alpha-numeric letters and any of the characters returned by keys:get-shifted-key-list.

  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 : (union 'left 'middle 'right)

  x : inexact
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 : (implements 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

See also Actions and completeness, section 27.2.

- `(test:number-pending-actions) ⇒ integer`

  Returns the number of simulated events that have started but not yet completed. An error that has been caught but not yet reraised counts as an incomplete action. Thus, a result of zero implies that there is no pending error and no action is still running.

test:reraise-error

See also Errors, section 27.3.

- `(test:reraise-error) ⇒ void`

  Reraises any pending error or else returns void.

test:run-interval

See also Actions and completeness, section 27.2.

- `(test:run-interval msec) ⇒ void`
  
  msec : number

  Sets the run interval to `msec` milliseconds.

- `(test:run-interval) ⇒ number`

  Returns the current setting for the number of milliseconds.

test:run-one

- `(test:run-one f) ⇒ void`
  
  f : (→ void)

  Runs the function \(f\) as if it was a simulated event. See also section 27.
test:set-check-box!

- (test:set-check-box! check-box state) ⇒ void
  
  check-box : (union (instance check-box%) string)
  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 (instance choice%) string)
  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 (instance radio-box%) string)
  state : boolean

  Clears the radio-box% item if state is #f, and sets it otherwise.
  If check-box is a string, this function searches for a radio-box% with a label matching that string, otherwise it uses radio-box itself.
This is the interface hierarchy for the editor and text classes in the framework.

```
editor<%>
|   |
editor:basic<%> (class->interface text<%>)
|   |
+----------------+-------------------+
|   |   |   |   |
editor:keymap<%> | editor:backup-autosave<%> | text:basic<%> | text:return<%> |
|   |   |   |   |
|   |   |   |   |
editor:autowrap<%> | editor:info<%> | text:clever-file-format<%>
|   |   |
+-------+------+------------+
|   |   |
editor:file<%> | text:searching<%> | text:info<%>
|   |   |
```

28.1 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-styles-fixed

If the result of this function is #t, the styles in this text<%> will be fixed. See also 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) ⇒ (-i void)
  start : exact-integer
  end : exact-integer
  color : (instance color<%>)
```
bitmap : (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%))
  Defaultly 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.

- (send a-text:basic set-styles-fixed fixed?) ⇒ void
  fixed? : boolean

28.2 text:basic-mixin

Domain: editor:basic<%>
28.2. text:basic-mixin

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.

- (make-object text:basic-mixin% line-spacing tabstops) ⇒ text:basic-mixin% object
  line-spacing = 1.0 : non-negative real number
  tabstops = null : list of real numbers

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.

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

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:basic-mixin after-change-style start len) ⇒ void
  start : exact non-negative integer
  len : exact non-negative integer

See set-styles-fixed.

after-insert

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

The start argument specifies the position of the insert. The len argument specifies the total length (in positions) of the inserted items.
on-change-style

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 section 8.8 (page 153)). Use after-change-style to modify the editor, if necessary.

See also on-edit-sequence.

- (send a-text:basic-mixin on-change-style start len) ⇒ void
  start : exact non-negative integer
  len : exact non-negative integer
  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.

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 section 8.8 (page 153)).

- (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)
  The before? argument is #t when the method is called just before a painting the contents of the editor or #f when it is called after painting. The left, top, right, and bottom arguments specify which region of the editor is being repainted, in editor coordinates. To get the coordinates for dc, offset editor coordinates by adding (dx, dy). See section 8.5 (page 152) for information about draw-caret.
  See also invalidate-bitmap-cache.
  Draws the rectangles installed by highlight-range.
28.3 text:searching

Extends: text:basic

Extends: editor:keymap

Any object matching this interface can be searched.

28.4 text:searching-mixin

Domain: text:basic

Domain: editor:keymap

Implements: text:searching

Implements: text:basic

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))
  
  Defaultly returns (list keymap:get-global)
  
  This returns a list containing the super-class’s keymaps, plus the result of keymap:get-search

28.5 text:return

Extends: (class->interface text)

Objects supporting this interface were created by text:return-mixin.

28.6 text:return-mixin

Domain: (class->interface text)
Implements: `text: return`

Use this buffer to perform some special action when `return` is typed.

```
- (make-object text:return-mixin% return) ⇒ text:return-mixin% object
  return : (λ 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
```

Either lets the keymap handle the event or calls `on-default-char`.

If `key` is either `return` or `newline`, only invoke the `return` thunk (initialization argument) and do nothing else.

### 28.7 `text: info`

Extends: `text: basic`

Objects supporting this interface are expected to send information about themselves to the frame that is displaying them.

### 28.8 `text: info-mixin`

Domain: `text: basic`

Domain: `editor: keymap`

Implements: `text: info`

Implements: `text: basic`

Implements: `editor: keymap`

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

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 end) ⇒ void
  
  start: exact non-negative integer
  end: exact non-negative integer

  The start argument specifies the starting position of the deleted range. The len argument specifies number of deleted items (so start + length is the endig position of the deleted range).

  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

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

  The start argument specifies the position of the insert. The len argument specifies the total length (in positions) of the inserted items.

  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

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.
28.9  text:clever-file-format

Extends: (class->interface text%)

Objects supporting this interface are expected to support a clever file format when saving.

28.10  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.

- (make-object text:clever-file-format-mixin% line-spacing tabstops) ⇒ text:clever-file-format-mixin% object

  line-spacing = 1.0 : non-negative real number
  tabstops = null : list of real numbers

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.
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

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 : string
  format : symbol in '(guess standard text text-force-cr same copy)

The filename argument is the name the file will be saved to. See load-file for information about format.

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.

28.11  text:basic% = (text:basic-mixin (editor:basic-mixin text%))

- (make-object text:basic% line-spacing tabstops) ⇒ text:basic% object
  line-spacing = 1.0 : non-negative real number
  tabstops = null : list of real numbers

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.

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.

28.12  text:keymap% = (editor:keymap-mixin text:basic%)

- (make-object text:keymap% line-spacing tabstops) ⇒ text:keymap% object
  line-spacing = 1.0 : non-negative real number
  tabstops = null : list of real numbers

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.

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.
28.13  text: return% = (text: return-mixin text: keymap%)

```
- (make-object text: return%  line-spacing  tabstops) => text: return% object
  line-spacing = 1.0  : non-negative real number
  tabstops = null  : list of real numbers
```

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.

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.

28.14  text: autowrap% = (editor: autowrap-mixin text: keymap%)

```
- (make-object text: autowrap%  line-spacing  tabstops) => text: autowrap% object
  line-spacing = 1.0  : non-negative real number
  tabstops = null  : list of real numbers
```

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.

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.

28.15  text: file% = (editor: file-mixin text: autowrap%)

```
- (make-object text: file%  line-spacing  tabstops) => text: file% object
  line-spacing = 1.0  : non-negative real number
  tabstops = null  : list of real numbers
```

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.

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.
28.16  text:clever-file-format% = (text:clever-file-format-mixin text:file%)

- (make-object text:clever-file-format% line-spacing tabstops) ⇒ text:clever-file-format% object
  line-spacing = 1.0 : non-negative real number  
  tabstops = null : list of real numbers

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.

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.

28.17  text:backup-autosave% = (editor:backup-autosave-mixin text:clever-file-format%)

- (make-object text:backup-autosave% line-spacing tabstops) ⇒ text:backup-autosave% object
  line-spacing = 1.0 : non-negative real number  
  tabstops = null : list of real numbers

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.

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.

28.18  text:searching% = (text:searching-mixin text:backup-autosave%)

- (make-object text:searching% line-spacing tabstops) ⇒ text:searching% object
  line-spacing = 1.0 : non-negative real number  
  tabstops = null : list of real numbers

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.

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.
28.19  \text{text:info\%} = (\text{text:info-mixin}\ (\text{editor:info-mixin}\ \text{text:searching\%}))

\text{text:info\%} = (\text{text:info-mixin}\ (\text{editor:info-mixin}\ \text{text:searching\%}))

- \text{(make-object text:info\% line-spacing tabstops)} \Rightarrow \text{text:info\% object}
  
  \text{line-spacing} = 1.0 \text{ : non-negative real number}
  
  \text{tabstops} = \text{null} \text{ : list of real numbers}

The \text{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 \text{set-tabs} for information about \text{tabstops}.

A new \text{keymap\%} object is created for the new editor. See also \text{get-keymap} and \text{set-keymap}.

A new \text{style-list} object is created for the new editor. See also \text{get-style-list} and \text{set-style-list}. 
29. Version

29.1 Version Utilities

version:add-spec

- (version:add-spec spec revision) ⇒ void
  spec : any scheme value
  revision : any scheme value

  These two values are appended to the version string. write is used to transform them to strings. For example:
  (version:add-version-spec 's 1)
  in version 45 will make the version string be "45s1". The symbols 'f and 'd are used internally for framework and drscheme revisions.

version:version

This function returns a string describing the version of this application. See also version:add-spec.

- (version:version) ⇒ string
Index

active-child, 110
add-tall-snip, 14
add-wide-snip, 14
after-change-style, 126, 140
after-delete, 127, 144
after-edit-sequence, 19, 127
after-insert, 127, 140, 144
after-load-file, 23
after-new-child, 37, 111
after-save-file, 23
after-set-position, 127, 144
after-set-size-constraint, 128
anchor-status-changed, 40
application:current-app-name, 10
autosave:register, 11
autosave?, 24
autosaving, 11
backup?, 24
“backward-kill-word”, 103
backward-sexp, 122
balance-parens, 122
balance-quotes, 122
begin-edit-sequence, 19
’border, 112
can-close-all?, 87
can-close?, 37, 77
can-remove-frame?, 87
canvas
scroll bars, 12, 15
canvas:basic-mixin, 12
canvas:basic<%>, 12
canvas:basic%, 14
canvas:info-mixin, 13
canvas:info<%>, 13
canvas:info%, 15
canvas:wide-snip-mixin, 14
canvas:wide-snip<%>, 13
canvas:wide-snip%, 15
“capitalize-word”, 103
“center-view-on-line”, 103
chain-to-keymap, 98
clear, 87
close, 34
“collapse-newline”, 103
“collapse-space”, 103
comment-out-selection, 122
container-size, 111, 112
contents, 108
’copy, 25, 146
“copy-click-region”, 103
“copy-clipboard”, 103
“cut-click-region”, 103
“cut-clipboard”, 103
delete, 108
determine-width, 38
do-autosave, 24
“do-macro”, 103
down-sexp, 123
“downcase-word”, 103
edit-menu:after-preferences, 44
edit-menu:between-clear-and-select-all, 44
edit-menu:between-copy-and-paste, 44
edit-menu:between-cut-and-copy, 44
edit-menu:between-find-and-preferences, 44
edit-menu:between-paste-and-clear, 44
edit-menu:between-redo-and-cut, 45
edit-menu:between-select-all-and-find, 45, 68
edit-menu:clear, 45
edit-menu:clear-help-string, 45
edit-menu:clear-on-demand, 45
edit-menu:clear-string, 46
edit-menu:copy, 46
edit-menu:copy-help-string, 46
edit-menu:copy-on-demand, 46
edit-menu:copy-string, 46
edit-menu:cut, 47
edit-menu:cut-help-string, 47
edit-menu:cut-on-demand, 47
edit-menu:cut-string, 47
edit-menu:find, 47, 74
edit-menu:find-again, 48, 74
edit-menu:find-again-help-string, 48
edit-menu:find-again-on-demand, 48
edit-menu:find-again-string, 48
edit-menu:find-help-string, 48
edit-menu:find-on-demand, 48
edit-menu:find-string, 49
edit-menu:get-clear-item, 49
edit-menu:get-copy-item, 49
edit-menu:get-cut-item, 49
edit-menu:get-find-again-item, 49
edit-menu:get-find-item, 49
edit-menu:get-paste-item, 49
edit-menu:get-preferences-item, 50
edit-menu:get-replace-and-find-again-item, 50
edit-menu:get-select-all-item, 50
edit-menu:get-undo-item, 50
edit-menu:paste, 50
edit-menu:paste-help-string, 50
edit-menu:paste-on-demand, 51
edit-menu:paste-string, 51
edit-menu:preferences, 51
edit-menu:preferences-help-string, 51
edit-menu:preferences-on-demand, 51
edit-menu:preferences-string, 52
edit-menu:redo, 52
edit-menu:redo-help-string, 52
edit-menu:redo-on-demand, 52
edit-menu:redo-string, 52
edit-menu:replace-and-find-again, 52, 74
edit-menu:replace-and-find-again-help-string, 53
edit-menu:replace-and-find-again-on-demand, 53
edit-menu:replace-and-find-again-string, 53
edit-menu:select-all, 53
edit-menu:select-all-help-string, 53
edit-menu:select-all-on-demand, 54
edit-menu:select-all-string, 54
edit-menu:undo, 54
edit-menu:undo-help-string, 54
edit-menu:undo-on-demand, 54
edit-menu:undo-string, 55
editing-this-file?, 17, 23
editor:autowrap-mixin, 22
editor:autowrap<%, 22
editor:backup-autosave-mixin, 25
editor:backup-autosave<%, 24
editor:basic-mixin, 18
editor:basic<%, 17
editor:file-mixin, 22
editor:file<%, 22
editor:info-mixin, 26
editor:info<%, 26
editor:keydown-mixin, 21
editor:keydown<%, 21
editors
events, 143
hooks, 19, 20, 25, 126–128, 140, 141, 144
locking, 20, 26
modified, 26
multiple changes, 19
end-edit-sequence, 19
“end-macro”, 103
exit callbacks, 27
exit:can-exit?, 27
exit:exit, 27
exit:frame-exiting, 27
exit:insert-can?-callback, 28
exit:insert-on-callback, 28
exit:on-exit, 28
exiting, 27
exn:unknown-preference, 119, 120
file-menu:after-quit, 55
file-menu:between-close-and-quit, 55
file-menu:between-new-and-open, 55
file-menu:between-open-and-revert, 55
file-menu:between-print-and-close, 55
file-menu:between-revert-and-save, 56
file-menu:between-save-as-and-print, 56
file-menu:close, 56
file-menu:close-help-string, 56
file-menu:close-on-demand, 56
file-menu:close-string, 57
file-menu:get-close-item, 57
file-menu:get-new-item, 57
file-menu:get-open-item, 57
file-menu:get-print-item, 57
file-menu:get-quit-item, 57
file-menu:get-revert-item, 57
file-menu:get-save-as-item, 58
file-menu:get-save-item, 58
file-menu:new, 58
file-menu:new-help-string, 58
file-menu:new-on-demand, 58
file-menu:new-string, 58
file-menu:open, 59
file-menu:open-help-string, 59
file-menu:open-on-demand, 59
file-menu:open-string, 59
file-menu:print, 59, 68
file-menu:print-help-string, 60
file-menu:print-on-demand, 60
file-menu:print-string, 60
file-menu:quit, 60
file-menu:quit-help-string, 60
file-menu:quit-on-demand, 60
file-menu:quit-string, 61
file-menu:revert, 61, 68
file-menu:revert-help-string, 61
file-menu:revert-on-demand, 61
file-menu:revert-string, 61
file-menu:save, 62, 69
file-menu:save-as, 62, 69
file-menu:save-as-help-string, 62
file-menu:save-as-on-demand, 62
INDEX

file-menu:save-as-string, 62
file-menu:save-help-string, 63
file-menu:save-on-demand, 63
file-menu:save-string, 63
files
  formats, 93
  loading, 23
  names, 23
  opening, 93
  saving, 23, 25, 146
  selecting, 30
find-down-sexp, 123
"find-string", 103
"find-string-replace", 103
"find-string-reverse", 103
find-up-sexp, 123
finder:common-get-file, 30
finder:common-get-file-list, 30
finder:common-put-file, 31
finder:current-find-file-directory, 31
finder:default-extension, 31
finder:dialog-parent-parameter, 32
finder:get-file, 32
finder:open-file, 102
finder:put-file, 32, 102
finder:std-get-file, 32
finder:std-put-file, 33
flash-backward-sexp, 123
flash-forward-sexp, 123
for-each-frame, 87
format handler, 93
forward-invalidate, 108
forward-sexp, 123
frame groups, 87
frame:label-changed, 88
frame:basic-mixin, 36
frame:basic<%>, 34
frame:basic%, 77
frame:editor-mixin, 68
frame:editor<%>, 66
frame:editor%, 81
frame:file-mixin, 76
frame:file<%>, 76
frame:info-mixin, 39
frame:info<%>, 38
frame:info%, 78
frame:pasteboard-info-file%, 85
frame:pasteboard-info-mixin, 42
frame:pasteboard-info<%>, 41
frame:pasteboard-info%, 79
frame:pasteboard-mixin, 71
frame:pasteboard<%>, 71
frame:pasteboard%, 84
frame:reorder-menus, 86
frame:searchable-mixin, 74
frame:searchable<%>, 72
frame:searchable%, 83
frame:standard-menus-mixin, 65
frame:standard-menus<%>, 42
frame:standard-menus%, 80
frame:text-info-file%, 83
frame:text-info-mixin, 41
frame:text-info<%>, 40
frame:text-info%, 78
frame:text-mixin, 71
frame:text<%>, 70
frame:text%, 82
get, 108
get-active-frame, 88
get-area-container, 35
get-area-container%, 35
get-backward-sexp, 123
get-canvas, 66
get-canvas<%>, 66
get-canvas%, 66
get-chained-keymaps, 97
get-checkable-menu-item%, 63
get-edit-menu, 63
get-editor, 67
get-editor<%>, 67, 71, 72, 75
get-editor%, 67, 71, 72, 75
get-entire-label, 67
get-file, 20
get-file-menu, 63
get-filename, 35, 69
get-forward-sexp, 123
get-frames, 88
get-help-menu, 63
get-info-canvas, 38
get-info-editor, 38
get-info-panel, 39
get-keymaps, 21, 23, 142
get-label, 69
get-label-prefix, 67
get-limit, 124
get-map-function-table, 97
get-map-function-table/ht, 97
get-mdi-parent, 88
get-menu-bar%, 35
get-menu-item%, 64
get-menu%, 64
get-string, 90
get-styles-fixed, 138
get-tab-size, 124
get-text-to-search, 72
get-top-level-window, 17
“goto-line”, 103
“goto-position”, 103
group: %, 87
group: get-the-frame-group, 89
’guess, 25, 146
gui-utils: cursor-delay, 90
gui-utils: delay-action, 90
gui-utils: get-choice, 91
gui-utils: get-clickback-delta, 91
gui-utils: get-clicked-clickback-delta, 91
gui-utils: local-busy-cursor, 91
gui-utils: next-untitled-name, 92
gui-utils: open-input-buffer, 92
gui-utils: read-snips/chars-from-text, 92
gui-utils: show-busy-cursor, 92
gui-utils: text-snip< %>, 90
gui-utils: unsaved-warning, 92
handler: edit-file, 93
handler: find-format-handler, 93
handler: find-named-format-handler, 93
handler: insert-format-handler, 94
handler: open-file, 94
has-focus?, 18
help-menu: about, 64, 70
help-menu: about-help-string, 64
help-menu: about-on-demand, 64
help-menu: about-string, 65, 70
help-menu: after-about, 65
help-menu: before-about, 65
help-menu: get-about-item, 65
’hide-hscroll, 12, 15
hide-search, 72
’hide-vscroll, 12, 15
highlight-parens, 124
highlight-range, 138
icon: get-anchor-bitmap, 95
icon: get-autowrap-bitmap, 95
icon: get-gc-off-bitmap, 95
icon: get-gc-on-bitmap, 95
icon: get-lock-bitmap, 95
icon: get-paren-highlight-bitmap, 95
icon: get-unlock-bitmap, 96
initial-autowrap-bitmap, 139
insert-frame, 88
insert-return, 124
invalidate, 109
key names, 98
keyboard focus
   notification, 13, 20, 38, 128
keyboard mapping, 97
keymap: aug-keymap-mixin, 97
keymap: call/text-keymap-initializer, 100
keymap: canonicalize-keybinding-string, 101
keymap: get-editor, 101
keymap: get-file, 101
keymap: get-global, 101
keymap: get-search, 101
keymap: make-meta-prefix-list, 102
keymap: send-map-function-meta, 102
keymap: setup-editor, 102
keymap: setup-file, 102
keymap: setup-global, 102
keymap: setup-search, 105
keymaps, 97
   chaining, 98
   in an editor, 116, 117, 128, 140, 145–149
keys: get-shifted-key-list, 106
“kill-word”, 103
“load-file”, 102
local-edit-sequence?, 18
locate-file, 88
lock, 20, 26
lock-status-changed, 39
locked?, 18
make-editor, 67
make-root-area-container, 35, 39, 75
map-function, 98
match-cache: %, 108, 114
matching cache, 108
max-count, 109
’mdi-child, 36, 77–85
’mdi-parent, 36, 77–85
Meta, 102
mouse mapping, 97
move-to-search-or-reverse-search, 72
move-to-search-or-search, 73
move/copy-to-edit, 139
’no-caption, 36, 77–85
’no-caret, 141
’no-hscroll, 12, 15
’no-resize-border, 36, 77–85
’no-system-menu, 36, 77–85
’no-vscroll, 12, 15
on-activate, 76
on-change, 25
on-change-style, 141
on-close, 18, 22, 25, 37, 40, 41, 70, 76
on-close-all, 88
on-drop-file, 37
on-edit-sequence, 20
on-focus, 13, 20, 38, 128
on-local-char, 143
on-new-box, 21
on-paint, 141
on-save-file, 25, 146
on-size, 14
on-subwindow-char, 65
“open-line”, 103
overwrite-status-changed, 41
panel:single-mixin, 110
panel:single-pane%, 112
panel:single-window-mixin, 111
panel:single-window<%>, 111
panel:single<%>, 110
panel:single%, 112
paren:backward-match, 113
paren:balanced?, 114
paren:forward-match, 114
paren:skip-whitespace, 114
parenthesis matching, 108, 113
“paste-click-region”, 103
“paste-clipboard”, 103
pasteboard:backup-autosave%, 116
pasteboard:basic%, 116
pasteboard:file%, 116
pasteboard:info%, 117
pasteboard:keydown%, 116
path-utils:generate-autosave-name, 118
path-utils:generate-backup-name, 118
place-children, 111
preferences, 119
preferences:add-callback, 119
preferences:add-font-panel, 119
preferences:add-general-panel, 119
preferences:add-panel, 119
preferences:get, 120
preferences:hide-dialog, 120
preferences:read, 120
preferences:restore-defaults, 120
preferences:save, 120
preferences:set, 120
preferences:set-default, 120
preferences:set-un/marshall, 121
preferences:show-dialog, 121
put, 109
put-file, 21
quiting, 27
remove-autosave, 24
remove-frame, 89
remove-parens-forward, 124
remove-sexp, 124
“remove-space”, 103
replace, 73
replace-all, 73
replace&search, 73
“ring-bell”, 102
run-after-edit-sequence, 18
’same, 25, 146
save-as, 67
“save-file”, 102
“save-file-as”, 102
scheme-paren:backward-containing-sexp, 130
scheme-paren:backward-match, 130
scheme-paren:balanced?, 130
scheme-paren:forward-match, 130
scheme-paren:get-comments, 131
scheme-paren:get-paren-pairs, 131
scheme-paren:get-quote-pairs, 131
scheme:add-preferences-panel, 128
scheme:get-keymap, 129
scheme:get-style-list, 129
scheme:get-wordbreak-map, 129
scheme:init-wordbreak-map, 129
scheme:setup-keymap, 129
scheme:text-mixin, 126
scheme:text<%>, 122
scheme:text%, 128
search-again, 73
select-backward-sexp, 124
“select-click-line”, 103
“select-click-word”, 103
select-down-sexp, 125
select-forward-sexp, 125
select-up-sexp, 125
set-active-frame, 89
set-anchor, 144
set-editor, 13
set-empty-callbacks, 89
set-filename, 23
set-info-canvas, 39
set-label, 70
set-label-prefix, 68
set-modified, 26
set-overwrite-mode, 145
set-search-direction, 73
set-styles-fixed, 139
set-tab-size, 125
’show-caret, 141
’show-inactive-caret, 141
splay, 109
’sstandard, 25, 146
“start-macro”", 103
style lists
INDEX

in an editor, 116, 117, 128, 140, 145-149

sum, 109

tabify, 125
tabify-all, 125
tabify-on-return?, 125
tabify-selection, 125
test:button-push, 134
test:close-top-level-window, 134
test:current-get-eventspaces, 135
test:keystroke, 135
test:menu-select, 135
test:mouse-click, 135

test:new-window, 136
test:number-pending-actions, 136
test:reraise-error, 136
test:run-interval, 136

test:run-one, 136
‘text, 25, 146
’text-force-cr, 25, 146

text:autowrap%, 147

text:backup-autosave%, 148

text:basic-mixin, 139

text:basic<%, 138

text:basic%, 146

text:clever-file-format-mixin, 145

text:clever-file-format<%, 145

text:clever-file-format%, 148

text:file%, 147

text:info-mixin, 143

text:info<%, 143

text:info%, 149

text:keymap%, 146

text:return-mixin, 142

text:return<%, 142

text:return%, 147

text:searching-mixin, 142

text:searching<%, 142

text:searching%, 148

times, 109
“toggle-anchor”, 103
“toggle-overwrite”, 103

toggle-search-focus, 74
“transpose-chars”, 103

transpose-sexp, 126
“transpose-words”, 103

uncomment-selection, 126
unhide-search, 74
up-sexp, 126
“upcase-word”, 103
update-info, 39, 41

version:add-spec, 150

156