API documentation for mint

Contents

1 Overview

Mobile interface toolkit. Mint library is a set of goodies for desktop applications. It allows applications to use common theme, provides standard dialogs and an implementation of event loop.

2 mint/config.eh — Mint preferences and reading of configuration files.

```
use "mint/config.eh"
```

2.1 Description

This header gives direct access to the Mint preferences and also provides function to read arbitrary configuration file.

The following preferences are available in the current version of Mint:

`iconTheme` Name of the icon theme.

`listIconSize` Size of icons used in lists and menus.

`dialogIconList` Size of icons used in dialog windows.

`dialogFont` Font used in dialog windows.

2.2 Types

```
type Config = {
  iconTheme: String = "",
  listIconSize: Int = 16,
  dialogIconSize: Int = 32,
  dialogFont: Int = FACE_SYSTEM | SIZE_MED | STYLE_PLAIN
}
```

Mint preferences. Use getConfig() function to obtain preferences for your system (stored in /cfg/mintprefs). If preferences file does not exist returned structure is initialized with default values (shown above).

2.3 Functions

```
def getConfig(): Config;
```

Returns Mint preferences structure. The structure can be modified.

```
def readCfgFile(file: String): Dict;
```

Reads configuration file. This function reads all strings in form `key=value` from the specified file and returns them in a dictionary. All strings that are not in this form are ignored.

3 mint/themeicon.eh — Icons from icon theme.

```
use "mint/themeicon.eh"
```

3.1 Description

Icon themes provide stock of icons to use in applications. Using themeIcon() function you can load one of standard icons or custom icon for preferred size. This function accepts two arguments - name and size. You can use special constants SIZE_LIST and SIZE_DIALOG to use sizes from preferences.

Note: Memory and storage on mobile device can be very limited so it is possible that no icon set is installed at all. In this case themeIcon() will return null, be sure to handle this case in your interface.

3.1.1 Where to install custom icons

If you want to provide just one icon, install it under `/res/icons`. For example

```
/res/icons/foo.png
```

This icon will be returned if you call `themeIcon("foo")`. If you want to provide several sizes of icons, install them under `/res/icons/size`. Like

```
/res/icons/16/foo.png
/res/icons/24/foo.png
/res/icons/32/foo.png
/res/icons/48/foo.png
```

It is recommended to stick to these four sizes.

Finally, if you want provide different icons for some icon themes, install them in corresponding theme folders (but be sure to install generic icons as well). If `themeIcon(name, size)` is called the sequence of locations in which icons are searched is the following:

/res/icons/*iconTheme*/*size*/*name*.png
/res/icons/*size*/*name*.png
/res/icons/*name*.png
/res/icons/*iconTheme*/*size*/image-missing.png

3.1.2 Standard icons

Icon themes currently provide 66 standard icons. You can use corresponding constants as arguments for themeIcon() to get them. Icons are the following:

File icons

FOLDER	for folders
FILE_BINARY	for binary file types and files with unknown content
FILE_AUDIO	for audio file types
FILE_EXECUTABLE	for executable file types
FILE_IMAGE	for image file types
FILE_INTERNET	for HTML pages, links and similar file types
FILE_PACKAGE	for archives, packages and similar file types
FILE_TEXT	for text file types
FILE_VIDEO	for video file types

Dialog choices

DIALOG_YES	for positive answer in dialog
DIALOG_NO	for negative answer in dialog
DIALOG_CLOSE	for closing dialog without taking any action

Informative dialog icons

DIALOG_ERROR	for error message dialogs
DIALOG_INFORMATION	for information message dialogs
DIALOG_QUESTION	for question type dialogs
DIALOG_WARNING	for warning message dialogs

Navigation icons

GO_UP	for ”go up” action
GO_DOWN	for ”go down” action
GO_TOP	for ”go to the top” action
GO_BOTTOM	for ”go to the bottom” action
GO_PREVIOUS	for ”go to the previous element” action
GO_NEXT	for ”go to the next element” action
GO_FIRST	for ”go to the first element” action
GO_LAST	Icon for ”go to the last element” action
GO_HOME	for ”go to the home location” action
GO_JUMP	for ”jump to the specified location” action

Media icons

MEDIA_EJECT	for the eject action of a media player or a file manager
MEDIA_PAUSE	for the pause action of a media player
MEDIA_RECORD	for the record action of a media player
MEDIA_SEEK_BACKWARD	for the seek backward action of a media player
MEDIA_SEEK_FORWARD	for the seek forward action of a media player
MEDIA_SKIP_BACKWARD	for the skip backward action of a media player
MEDIA_SKIP_FORWARD	for the skip forward action of a media player
MEDIA_START	for the start or play action of a media player
MEDIA_STOP	for the stop action of a media player

Document related icons

DOCUMENT_NEW	for "create new document" action
FOLDER_NEW	for "create new folder" action
DOCUMENT_OPEN	for "open existing document" action
DOCUMENT_PREVIEW	for "preview document" action
DOCUMENT_RECENT	for "open recent document" action
DOCUMENT_SAVE	for "save current document" action
DOCUMENT_SAVE_AS	for "save current document in a new location" action

View related icons

VIEW_FULLSCREEN	for "view fullscreen" action
VIEW_REFRESH	for "refresh view" action
VIEW_RESTORE	for "restore view from fullscreen" action
ZOOM_IN	for "zoom in" action
ZOOM_OUT	for "zoom out" action
ZOOM_ORIGINAL	for "restore original zoom" action

Action icons

EDIT_COPY	for "copy" action
EDIT_CUT	for "cut" action
EDIT_DELETE	for "delete" action
EDIT_FIND	for "find" action
EDIT_PASTE	for "paste" action
EDIT_REDO	for "redo" action
EDIT_RENAME	for "rename" action
EDIT_REPLACE	for "replace" action
EDIT_UNDO	for "undo" action
LIST_ADD	for "add element to the list" action
LIST_REMOVE	for "remove element from the list" action
APP_EXIT	for "exit from application" action

Other icons

APP_CONFIGURE	for settings dialogs, etc.
APP_INTERNET	for actions or applications that go to the internet
APP_MEDIA	for media actions or applications
APP_PLUGIN	for managing packages or plugins
APP_TERMINAL	for launching terminal or something in terminal
IMAGE_MISSING	used when another icon is not found

3.2 Constants

```
const SIZE_LIST = 0;
```

Constant to use in themeIcon() for list icons. The size will be the same as returned by Config.listIconSize.

```
const SIZE_DIALOG = -1;
```

Constant to use in themeIcon() for dialog icons. The size will be the same as returned by Config.dialogIconSize.

```
const IMAGE_MISSING = "image-missing";
```

Icon used as a placeholder when some another icon is not found.

```
const FOLDER = "folder";
```

Icon for folders in the file system or other hierarchical groups.

```
const FILE_BINARY = "file-binary";
```

Icon for binary file types and files with unknown content.

```
const FILE_AUDIO = "file-audio";
```

Icon for audio file type.

```
const FILE_EXECUTABLE = "file-executable";
```

Icon for executable file types.

```
const FILE_IMAGE = "file-image";
```

Icon for image file types.

```
const FILE_INTERNET = "file-internet";
```

Icon for HTML pages, links and similar file types.

```
const FILE_PACKAGE = "file-package";
```

Icon for archives, packages and similar file types.

```
const FILE_TEXT = "file-text";
```

Icon for text file types.

```
const FILE_VIDEO = "file-video";
```

Icon for video file types.

```
const DIALOG_YES = "dialog-yes";
```

Icon for positive answer in dialog.

```
const DIALOG_NO = "dialog-no";
```

Icon for negative answer in dialog.

```
const DIALOG_CLOSE = "dialog-close";
```

Icon for closing dialog without taking any action.

```
const DIALOG_ERROR = "dialog-error";
```

Icon for error message dialogs.

```
const DIALOG_INFORMATION = "dialog-information";
```

Icon for information message dialogs.

```
const DIALOG_QUESTION = "dialog-question";
```

Icon for question type dialogs.

```
const DIALOG_WARNING = "dialog-warning";
```

Icon for warning message dialogs.

```
const GO_UP = "go-up";
```

Icon for ”go up” action.

```
const GO_DOWN = "go-down";
```

Icon for ”go down” action.

```
const GO_TOP = "go-top";
```

Icon for ”go to the top” action.

```
const GO_BOTTOM = "go-bottom";
```

Icon for ”go to the bottom” action.

```
const GO_PREVIOUS = "go-previous";
```

Icon for ”go to the previous element” action.

```
const GO_NEXT = "go-next";
```

Icon for ”go to the next element” action.

```
const GO_FIRST = "go-first";
```

Icon for ”go to the first element” action.

```
const GO_LAST = "go-last";
```

Icon for ”go to the last element” action.

```
const GO_HOME = "go-home";
```

Icon for ”go to the home location” action.

```
const GO_JUMP = "go-jump";
```

Icon for ”jump to the specified location” action.

```
const MEDIA_EJECT = "media-eject";
```

Icon for the eject action of a media player or a file manager.

```
const MEDIA_PAUSE = "media-pause";
```

Icon for the pause action of a media player.

```
const MEDIA_RECORD = "media-record";
```

Icon for the record action of a media player.

```
const MEDIA_SEEK_BACKWARD = "media-seek-backward";
```

Icon for the seek backward action of a media player.

```
const MEDIA_SEEK_FORWARD = "media-seek-forward";
```

Icon for the seek forward action of a media player.

```
const MEDIA_SKIP_BACKWARD = "media-skip-backward";
```

Icon for the skip backward action of a media player.

```
const MEDIA_SKIP_FORWARD = "media-skip-forward";
```

Icon for the skip forward action of a media player.

```
const MEDIA_START = "media-start";
```

Icon for the start or play action of a media player.

```
const MEDIA_STOP = "media-stop";
```

Icon for the stop action of a media player.

```
const DOCUMENT_NEW = "document-new";
```

Icon for "create new document" action.

```
const FOLDER_NEW = "folder-new";
```

Icon for "create new folder" action.

```
const DOCUMENT_OPEN = "document-open";
```

Icon for "open existing document" action.

```
const DOCUMENT_PREVIEW = "document-preview";
```

Icon for "preview document" action.

```
const DOCUMENT_RECENT = "document-recent";
```

Icon for "open recent document" action.

```
const DOCUMENT_SAVE = "document-save";
```

Icon for "save current document" action.

```
const DOCUMENT_SAVE_AS = "document-save-as";
```

Icon for "save current document in a new location" action.

```
const VIEW_FULLSCREEN = "view-fullscreen";
```

Icon for "view fullscreen" action.

```
const VIEW_REFRESH = "view-refresh";
```

Icon for "refresh view" action.

```
const VIEW_RESTORE = "view-restore";
```

Icon for "restore view from fullscreen" action.

```
const ZOOM_IN = "zoom-in";
```

Icon for ”zoom in” action.

```
const ZOOM_OUT = "zoom-out";
```

Icon for ”zoom out” action.

```
const ZOOM_ORIGINAL = "zoom-original";
```

Icon for ”restore original zoom” action.

```
const EDIT_COPY = "edit-copy";
```

Icon for ”copy” action.

```
const EDIT_CUT = "edit-cut";
```

Icon for ”cut” action.

```
const EDIT_DELETE = "edit-delete";
```

Icon for ”delete” action.

```
const EDIT_FIND = "edit-find";
```

Icon for ”find” action.

```
const EDIT_PASTE = "edit-paste";
```

Icon for ”paste” action.

```
const EDIT_REDO = "edit-redo";
```

Icon for ”redo” action.

```
const EDIT_RENAME = "edit-rename";
```

Icon for ”rename” action.

```
const EDIT_REPLACE = "edit-replace";
```

Icon for ”replace” action.

```
const EDIT_UNDO = "edit-undo";
```

Icon for ”undo” action.

```
const LIST_ADD = "list-add";
```

Icon for ”add element to the list” action.

```
const LIST_REMOVE = "list-remove";
```

Icon for "remove element from the list" action.

```
const APP_EXIT = "app-exit";
```

Icon for "exit from application" action.

```
const APP_CONFIGURE = "app-configure";
```

Icon for launching application settings.

```
const APP_INTERNET = "app-internet";
```

Icon for action or application that uses internet.

```
const APP_MEDIA = "app-media";
```

Icon for audio/video applications or actions.

```
const APP_PLUGIN = "app-plugin";
```

Icon for applications or menus that manage packages or plugins.

```
const APP_TERMINAL = "app-terminal";
```

Icon for terminal actions or applications.

3.3 Functions

```
def themeIcon(name: String, size: Int = SIZE_LIST): Image;
```

Returns icon from icon theme. If there is no icon for given size then closest size is returned. Icons are NOT rescaled. If there is no icon for given name then IMAGE_MISSING icon is returned. If IMAGE_MISSING icon is missing as well, null is returned. Constants `SIZE_LIST` and `SIZE_DIALOG` may be specified to use sizes from settings.

4 mint/dialog.eh — Common dialogs.

```
use "mint/dialog.eh"
```

4.1 Description

This header provides a set of dialog screens to use in applications. Dialog is a modal screen that shows up, requests information, returns requested value and turns back to the previous screen. Many of dialogs may contain icons which are specified as string identifiers. The icon that dialog will contain is exactly as returned by

```
themeIcon(name, SIZE_DIALOG)
```

4.2 Functions

```
def Screen.run(): Menu;
```

Runs modal screen. This method can be used to implement custom dialogs. It shows the screen and blocks until menu item is chosen. Then previous screen of application is restored and chosen menu item is returned.

```
def showInfo(title: String, msg: String, timeout: Int = 0);
```

Shows information message. This function presents dialog with a text message and DIALOG_INFORMATION icon. If *timeout* is specified and positive then the dialog will automatically disappear after specified number of milliseconds.

```
def showWarning(title: String, msg: String, timeout: Int = 0);
```

Shows warning message. This function presents dialog with a text message and DIALOG_WARNING icon. If *timeout* is specified and positive then the dialog will automatically disappear after the specified number of milliseconds.

```
def showError(title: String, msg: String, timeout: Int = 0);
```

Shows error message. This function presents dialog with a text message and DIALOG_ERROR icon. If *timeout* is specified and positive then the dialog will automatically disappear after the specified number of milliseconds.

```
def showMessage(title: String, msg: String, icon: String = null,
        timeout: Int = 0);
```

Shows message dialog. This function presents dialog with a text message, optional icon and "Close" command. If *timeout* is specified and positive then the dialog will automatically disappear after the specified number of milliseconds.

```
def showYesNo(title: String, msg: String, y: String = "Yes",
        n: String = "No"): Bool;
```

Shows question dialog with two variants. This function presents dialog with a text message and two commands which by default are "Yes" and "No". The function returns true if the first command is chosen and false otherwise.

```
def showOption(title: String, msg: String, variants: [String],
        icon: String = null): Int;
```

Shows question dialog with specified list of options. This function presents dialog with a text message and a set of menu items. Captions of menu items are read from the *variants* array. The function returns index of chosen variant in range from `0` to `variants.len-1`.

```
def showList(title: String, lines: [String],
        icons: [String] = null): Int;
```

Shows dialog with a list of choices. This function presents ListBox with given text strings and icons. If *icons* array is not null then it must be the same length as *lines*. The dialog has two menu commands: "Ok" and "Cancel". The function returns index of selected element or -1 if Cancel is pressed.

```
def showInput(title: String, msg: String = "", text: String = "",
        mode: Int = EDIT_ANY, maxsize: Int = 50): String;
```

Shows dialog with an input field. This function presents dialog with a message and an input field. Argument *mode* represents kind of information that can be entered in the input field and must be one of constants defined in ui_edit.eh. The dialog has two menu commands: "Ok" and "Cancel". The function returns entered string or null if Cancel is pressed.

```
def showFontDialog(title: String, font: Int = 0): Int;
```

Shows dialog which allows to choose font. The dialog has two menu commands: "Ok" and "Cancel". The function returns font specifier (see font.eh) or -1 if Cancel is pressed.

```
def showColorDialog(title: String, color: Int = 0): Int;
```

Shows dialog which allows to choose color. The dialog has two menu commands: "Ok" and "Cancel". The function returns color as 32-bit integer in form 0x00RRGGBB or -1 if Cancel is pressed.

```
def showFolderDialog(title: String, dir: String = null): String;
```

Shows dialog which allows to choose directory. The dialog has two menu commands: "Ok" and "Cancel" and also allows to create new dialog. The function returns path to the directory or null if Cancel is pressed.

```
def showOpenFileDialog(title: String, dir: String = null,
        filters: [String] = null): String;
```

Shows dialog which allows to choose existing file. The dialog has two menu commands: "Ok" and "Cancel". It only allows to choose existing file. The function returns path to the file or null if Cancel is pressed.

```
def showSaveFileDialog(title: String, dir: String = null,
        filters: [String] = null): String;
```

Shows dialog which allows to choose or create file. The dialog has two menu commands: "Ok" and "Cancel". It also allows creating new files and asks permission to overwrite if existing file is chosen. The function returns path to the file or null if Cancel is pressed.

5 mint/eventloop.eh — Event handling for graphical applications.

```
use "mint/eventloop.eh"
```

5.1 Description

This header provides automated handling of UI events. To use it just assign needed action to the desired event and start a loop. The following example shows this approach.

```
/* MINT hello world. */
use "mint/eventloop"
use "stdscreens"
use "ui"

def main(args: [String]) {
  // create screen
  var msgbox = new MsgBox("Hello, world!")
  // create event loop for it
  var loop = new EventLoop(msgbox)
  // add menu item to exit
```

```
  loop.onMenu(new Menu("Ok", 1), loop.quit)
  // start processing of events
  loop.start()
}
```

5.2 Types

```
type EventLoop < Any;
```

Event processing loop.

5.3 Functions

```
def EventLoop.new(scr: Screen);
```

Creates new event loop for specified screen.

```
def EventLoop.start();
```

Starts this event loop. This method shows screen that was used in constructor of this loop and blocks until quit() method is called.

```
def EventLoop.quit();
```

Stops this event loop. This method hides also hides the screen.

```
def EventLoop.onShow(handler: ());
```

Add action when screen shows or gains focus. If argument is null then action is removed.

```
def EventLoop.onHide(handler: ());
```

Add action when screen hides or loses focus. If argument is null then action is removed.

```
def EventLoop.onMenu(menu: Menu, handler: ());
```

Add menu element and the corresponding action to the screen. If argument is null then specified menu element is removed.

```
def EventLoop.onItem(item: Item, handler: ());
```

Add action when interactive form item is activated. Interactive form items are HyperlinkItem and HyperimageItem. If argument is null then action is removed.

```
def EventLoop.onStateChange(item: Item, handler: ());
```

Add action when a form item is edited by user. If argument is null then action is removed.

```
def EventLoop.onKeyPress(handler: (Int));
```

Add action when key is pressed. Argument of handler receives key code. If argument is null then action is removed.

```
def EventLoop.onKeyRelease(handler: (Int));
```

Add action when key is released. Argument of handler receives key code. If argument is null then action is removed.

```
def EventLoop.onKeyHold(handler: (Int));
```

Add action when key is held. Argument of handler receives key code. If argument is null then action is removed.

```
def EventLoop.onPtrPress(handler: (Int, Int));
```

Add action when screen is touched. Argument of handler receives touch coordinates. If argument is null then action is removed.

```
def EventLoop.onPtrDrag(handler: (Int, Int));
```

Add action when screen is touched and dragged. Argument of handler receives drag coordinates. If argument is null then action is removed.

```
def EventLoop.onPtrRelease(handler: (Int, Int));
```

Add action when touch is removed from screen. Argument of handler receives coordinates. If argument is null then action is removed.

6 mint/actionlist.eh — List of actions to implement program menus.

```
use "mint/actionlist.eh"
```

6.1 Description

This header provides ActionList screen which can be used to implement application menus. This screen presents set of choices to the user and calls action assigned to chosen element.

6.2 Types

```
type ActionList < Any;
```

Screen that represents list of actions.

6.3 Functions

```
def ActionList.new(title: String);
```

Creates new ActionList with specified title.

```
def ActionList.start(useCancel: Bool = true);
```

Shows list to the user and waits for response. When user selects element, the list closes and the corresponding action is executed. If *useCancel* is true then the list also has "Cancel" command which allows to close it without taking any action.

```
def ActionList.clear();
```

Removes all elements from this list.

```
def ActionList.add(text: String, icon: String, action: ());
```

Adds element to the end of this list. Icon name is a string following the specification of a mint/themeicon.eh header.

```
def ActionList.set(index: Int, text: String, icon: String, action: ());
```

Sets new value to the element of this list. Icon name is a string following the specification of a mint/themeicon.eh header.