NSApplication

Inherits From:	NSResponder : NSObject
Declared In:	appkit/NSApplication.h appkit/NSColorPanel.h appkit/NSDataLinkPanel.h appkit/NSHelpPanel.h appkit/NSPageLayout.h

Creating the Instance of NSApplication

+ (NSApplication *)**sharedApplication**	Returns the NSApplication instance, creating it if it doesn't yet exist.

Changing the Active Application

- (void)**activateIgnoringOtherApps:**(BOOL)*flag*	Makes this the active application. If *flag* is NO, the application is activated only if no other application is currently active.
- (void)**deactivate**	Deactivates the application.
- (BOOL)**isActive**	Returns whether this is the active application.

Running the Event Loop

- (void)**abortModal**	Aborts the event loop started by **runModalForWindow:**.
- (NSModalSession)**beginModalSessionForWindow:**(NSWindow *)*theWindow*	Sets up a modal session with *theWindow*.
- (void)**endModalSession:**(NSModalSession)*session*	

	Finishes a modal session.
- (BOOL)**isRunning**	Returns whether the main event loop is running.
- (void)**run**	Starts the main event loop.
- (int)**runModalForWindow:**(NSWindow *)*theWindow*	Starts a modal event loop for *theWindow*.
- (int)**runModalSession:**(NSModalSession)*session*	Runs a modal session.
- (void)**sendEvent:**(NSEvent *)*theEvent*	Dispatches events to other objects.
- (void)**stop:**(id)*sender*	Stops the main event loop.
- (void)**stopModal**	Stops the modal event loop.
- (void)**stopModalWithCode:**(int)*returnCode*	Stops the event loop started by **runModalForWindow:** and sets the code that **runModalForWindow:** will return.

Getting, Removing, and Posting Events

- (NSEvent *)**currentEvent**	Returns the current event.
- (void)**discardEventsForApplicationMatchingMask:**(unsigned int)*mask* **beforeEvent:**(NSEvent *)*lastEvent*	Removes from the event queue all events matching *mask* that were generated before *lastEvent.*
- (NSEvent *)**nextEventForApplicationMatchingMask:**(unsigned int)*mask* **untilDate:**(NSDate *)*expiration* **inMode:**(NSString *)*mode* **dequeue:**(BOOL)*flag*;	Returns the next event matching *mask*, or **nil** if no such event is found before the *expiration* date. If *flag* is YES, the event is removed from the queue. The *mode* argument names an NSRunLoop mode that determines what other ports are listened to and what timers may fire while the NSApplication is waiting for the event.
- (void)**postEventForApplication:**(NSEvent *)*event* **atStart:**(BOOL)*flag*	Adds *event* to the beginning of the application's event queue if *flag* is YES, and to the end otherwise.

Sending Action Messages

- (BOOL)**sendAction:**(SEL)*aSelector*	Sends an action message to *aTarget* or up the responder

to:(id)*aTarget* **from:**(id)*sender*	chain.
- (id)**targetForAction:**(SEL)*aSelector*	Returns the object that receives the action message *aSelector*.
- (BOOL)**tryToPerform:**(SEL)*aSelector* **with:**(id)*anObject*	Attempts to send a message to the application or the delegate.

Hiding All Windows

- (void)**hide:**(id)*sender*	Hides all the application's windows.
- (BOOL)**isHidden**	Returns YES if windows are hidden.
- (void)**unhide:**(id)*sender*	Restores hidden windows to the screen.
- (void)**unhideWithoutActivation**	Restores hidden windows without activating their owner.

Managing Windows

- (NSWindow *)**keyWindow**	Returns the the key window.
- (NSWindow *)**mainWindow**	Returns the main window.
- (NSWindow *)**makeWindowsPerform:**(SEL)*aSelector* **inOrder:**(BOOL)*flag*	Sends the *aSelector* message to the application's NSWindowsÐin front-to-back order if *flag* is YES, otherwise in the order of the array that the **windows** method returns.
- (void)**miniaturizeAll:**(id)*sender*	Miniaturizes all the receiver's application windows.
- (void)**preventWindowOrdering**	Suppresses the usual window ordering in handling the most recent mouse-down event.
- (void)**updateWindows**	Sends update message to all on-screen NSWindows.
- (NSArray *)**windows**	Returns an array of the application's NSWindows.
- (NSWindow *)**windowWithWindowNumber:**(int)*windowNum*	Returns the NSWindow object corresponding to *windowNum*.

Showing Standard Panels

- (void)**orderFrontColorPanel:**(id)*sender*	Brings up the color panel.
- (void)**orderFrontDataLinkPanel:**(id)*sender*	Shows the shared instance of the data link panel, creating it first if

	necessary.
- (void)**orderFrontHelpPanel:**(id)*sender*	Shows the application's help panel or the default one.
- (void)**runPageLayout:**(id)*sender*	Runs the application's page layout panel.

Getting the Main Menu

- (NSMenu *)**mainMenu**	Returns the **id** of the application's main menu.
- (void)**setMainMenu:**(NSMenu *)*aMenu*	Makes *aMenu* the application's main menu.

Managing the Windows Menu

- (void)**addWindowsItem:**(id)*aWindow* **title:**(NSString *)*aString* **filename:**(BOOL)*isFilename*	Adds a Windows menu item for *aWindow*.
- (void)**arrangeInFront:**(id)*sender*	Orders all registered NSWindows to the front.
- (void)**changeWindowsItem:**(id)*aWindow* **title:**(NSString *)*aString* **filename:**(BOOL)*isFilename*	Changes the Windows menu item for *aWindow*.
-(void)**removeWindowsItem:**(id)*aWindow*	Removes the Windows menu item for *aWindow.*
- (void)**setWindowsMenu:**(id)*aMenu*	Sets the Windows menu.
- (void)**updateWindowsItem:**(id)*aWindow*	Updates the Windows menu item for *aWindow*.
- (NSMenu *)**windowsMenu**	Returns the Windows menu.

Managing the Services menu

- (void)**registerServicesMenuSendTypes:**(NSArray *)*sendTypes* **returnTypes:**(NSArray *)*returnTypes*	Registers pasteboard types the application can send and receive.
- (NSMenu *)**servicesMenu**	Returns the Services menu.
- (void)**setServicesMenu:**(NSMenu *)*aMenu*	Sets the Services menu.
- (id)**validRequestorForSendType:**(NSString *)*sendType* **returnType:**(NSString *)*returnType*	Indicates whether the NSApplication can send and receive the specified types.

Getting the Display PostScript Context

- (NSDPSServerContext *)**context**	Returns the NSApplication's DPS context.

Reporting an Exception

- (void)**reportException:**(NSException *)*anException*	Logs the given exception by calling **NSLog()**.

Terminating the Application

- (void)**terminate:**(id)*sender*	Frees the NSApplication object and exits the application.

Assigning a Delegate

- (id)**delegate**	Returns the NSApplication's delegate.
- (void)**setDelegate:**(id)*anObject*	Makes *anObject* the NSApplication's delegate.

Implemented by the Delegate

- (NSDataLinkManager *)**application:**(id)*sender* **openFileWithoutUI:**(NSString *)*filename* **withType:**(NSString *)*aType*	Opens the specified file to run without a user interface. Work with the file will be under programmatic control of *sender*, rather than under keyboard control of the user. Although a file's type may by convention be reflected in its name, *aType* must be specified, and *filename* should not exclude the extension.
- (int)**application:**(NSApplication *)*sender* **openFile:**(NSString *)*filename* **withType:**(NSString *)*aType*	Like **application:openFileWithoutUI:withType:**, but brings up the user interface of the file's application, and returns YES or NO to indicate whether the file was successfully opened.
- (int)**application:**(NSApplication *)*sender* **openTempFile:**(NSString *)*filename* **withType:**(NSString *)*aType*	Like **application:openFile:withType:**, but a file opened through this method is assumed to be temporary; it's the application's responsibility to remove the file at the appropriate time.
- (BOOL)**applicationShouldTerminate:**(id)*sender*	Returns YES if the application should terminate.

Implemented by Observers

- (void)**applicationDidBecomeActive:**(NSNotification *)*notification*
  Invoked when the application has been activated.
- (void)**applicationDidHide:**(NSNotification *)*notification*
  Invoked when the application has been hidden.
- (void)**applicationDidInitialize:**(NSNotification *)*notification*
  Invoked before the application gets its first event.
- (void)**applicationDidResignActive:**(NSNotification *)*notification*
  Invoked when the application has been deactivated.
- (void)**applicationDidUnhide:**(NSNotification *)*notification*
  Invoked when the application has been unhidden.
- (void)**applicationDidUpdate:**(NSNotification *)*notification*
  Invoked when the application has updated its windows.
- (void)**applicationWillBecomeActive:**(NSNotification *)*notification*
  Invoked when the application is about to be activated.
- (void)**applicationWillHide:**(NSNotification *)*notification*
  Invoked when the application is about to be hidden.
- (void)**applicationWillInitialize:**(NSNotification *)*notification*
  Invoked before initializing the application.
- (void)**applicationWillResignActive:**(NSNotification *)*notification*
  Invoked when the application is about to be deactivated.
- (void)**applicationWillUpdate:**(NSNotification *)*notification*
  Invoked before the application updates its windows.