FCError

Inherits From: Object

Declared In: FCError.h

Class Description

The FCError class announces messages to the user. A number of standard messages are provided, as well as standard methods used in announcing messages: using these helps to provide the user with a consistent interface. A default announcement method can be registered for each class which uses the FCError class, making it possible to completely change the user interface of error handling in a class without changing each line where those errors are reported.

Each instance of FCError is associated with a particular header string (set using the **setHeaderString:** method) and a **printf()** -style message string (set with **setMessageString:**). The parameters to the messageString template can be filled in by calling the

completeMessageString: routine. Each FCError also stores a preferred method of announcing its message, which is used whenever the default **announce** method is called. New instances can be registered in the class' error registry via the **registerError:owner:** method, and recalled via the **registeredError:owner:** method.

There are quite a few pre-registered FCError's (which can be accessed via the **registeredError:** *name* method). They are:

noError
Header String: No Error
Message String: This is not an error.
test
Header String: Message System Test
Message String: This error message should be used for testing only.
message
Header String: Message
Message String: %s
Parameters: "Description of Bug."
bug
Header String: Program Bug
Message String: Program bug at line %d of file %s.
Parameters: **__LINE__** , **__FILE__**
bugDescription Header String: Program Bug
Message String: Program bug at line %d of file %s. %s
Parameters: **__LINE__** , **__FILE__** , "Description of Bug."
bugBadParameter
Header String: Program Bug

Message String: Program bug at line %d of file %s. Invalid parameter.
Parameters: **__LINE__ , __FILE__**

bugBadMethod Header String: Program Bug
Message String: Program bug at line %d of file %s. Invalid method call.
Parameters: **__LINE__ , __FILE__**

bugNotImplemented
Header String: Program Bug
Message String: Program bug at line %d of file %s. Invalid method call.
Parameters: **__LINE__ , __FILE__**

bugSubclassResponsibility
Header String: Program Bug
Message String: Program bug at line %d of file %s. Invalid method call.
Parameters: **__LINE__ , __FILE__**

bugBadFreeParm
Header String: Program Bug
Message String: Program bug at line %d of file %s. Invalid parameter to free.
Parameters: **__LINE__ , __FILE__**

bugCustomAssertionFailed
Header String: Program Bug
Message String: %s%d
Parameters: "Description of Bug."

bugDefaultAssertionFailed
Header String: Program Bug
Message String: %s%d
Parameters: "Description of Bug."

fopen
Header String: Program Bug

Message String: Problem opening file *%s* encountered at line %d of file %s.
Parameters: "FileName", **__LINE__** , **__FILE__**

Currently, FCError supports messages used for teaching, file system interface, bugs, or design rules. Possibilities for future extension include facilities for handling interactive dialogs such as save on quit, save on close, merge nets, etc.

A common use of FCError is to perform assertion tests. The macros **FC_ASSERTION()** , **FC_PRECONDITION()** , **FC_POSTCONDITION()** , and **FC_CLAIM()** are provided for this purpose. **FC_CLAIM()**'s tests can be disabled by compiling with **-D** FC_BLOCKCLAIMS, making it a particularly convenient macro for debugging purposes.

Instance Variables

Inherited from Object
None declared in this class.

Declared in FCError
FCString ***_fc_owner** ;
FCString ***_fc_handle** ;
FCString ***_fc_header** ;
FCString ***_fc_template** ;
FCString ***_fc_completeMessage** ;
SEL **_fc_preferredAnnouncer** ;
id **_fc_delegate** ;

_fc_owner	The name of the owner
_fc_handle	The registered name
_fc_header	"Error", "Warning", "Message", etc.
_fc_template	The message string
_fc_completeMessage	Parameters for the message string
_fc_preferredAnnouncer	The preferred announcer method
_fc_delegate	The delegate

Method Types

Factory Methods	+zone +alloc +initialize
Initializing	-init -initWithHeaderString:messageString:
Freeing	-free
Copying	-copyFromZone:
Setting and Querying the Default Announcers	+setDefaultAnnouncer:

	+defaultAnnouncer
	+setAnnouncer:forFile:
	+announcerForFile:
Setting and Querying the Error	-setMessage:
	-message
	-setMessageString:
	-messageString
	-completeMessageString:...
	-completeMessage
	-completeMessageString
	-setHeader:
	-header
	-setHeaderString:
	-headerString
Setting and Querying the Delegate	-setDelegate:
	-delegate
Setting and Querying the Preferred Announcer	
	-setPreferredAnnouncer:
	-preferredAnnouncer
Announcers	-announce
	-announceByExiting
	-announceByRaisingException
	-announceToStderr
	-announceWithPanel
	-announceWithSyslog

	-announceWithLogError -announceWithSound -announceToDelegate
Error registry	-registerError:owner: -deregister +registeredError:owner: +registeredError: -registeredErrorString -registeredOwnerString
Archiving	-write: -read:

Class Methods

alloc

+ **alloc;**

Returns a new instance of FCError. Memory for the new object is allocated from the memory zone returned by the FCError **zone** class method, localizing the FCError objects.

See also: **+ zone, + allocFromZone** (Object)

announcerForFile:

+ (SEL)**announcerForFile:**(const char *)*fileName;*

Gets the announcer method associated with a given file name. This announcer will be used for assertions that fail in that file.

defaultAnnouncer

+ (SEL)**defaultAnnouncer;**

Gets the default announcer method used by instances of FCError that do not have a Preferred Announcer.

initialize

+ **initialize;**

Initializes the class: registers the error reporting function that will be used to report exceptions raised by FCError, allocates the registration and file tables, and registers the standard, pre-registered errors.

Never invoke this method directly; it's sent for you when the application starts. Returns **self**.

registeredError:

+ **registeredError:**(const char *)*theError;*

Returns the standard FCError instance pre-registered by FCError under the name *theError*. Returns **nil** if no error is registered for the given key. The pre-registered errors are listed in the class description.

registeredError:owner:

+ **registeredError:**(const char *)*theError* **owner:**(const char *)*theOwner;*

Returns the FCError instance registered under the name *theError* for the owner *theOwner*. Returns **nil** if no such error is registered.

setAnnouncer:forFile:

+ **setAnnouncer:**(SEL)*theAnnouncer* **forFile:**(const char *)*fileName;*

Sets the announcer method associated with a given file name. This announcer will be used for assertions that fail in the file.

setDefaultAnnouncer:

+ **setDefaultAnnouncer:**(SEL)*theAnnouncer;*

Sets the default announcer method used by instances of FCError that do not have a Preferred

Announcer.

zone

+ (NXZone *)**zone;**

Returns the class zone. The **alloc** method allocates new instances of FCError from this zone of memory.

See also: **+ alloc**

Instance Methods

announce

- **announce;**

Calls the preferred announcer method of the FCError object. If the preferred announcer is set to FC_DEFAULT_ANNOUNCER, uses the default announcer for the class. Returns what the preferred announcer returns.

See also: **- preferredAnnouncer, + defaultAnnouncer**

announceByExiting

- **announceByExiting;**

Exits the application (returning status code 1). Never returns.

See also: **- announce**

announceByRaisingException

- **announceByRaisingException;**

Alerts the appropriate error handler that an error has occurred. Never returns.

See also: **- NX_RAISE(), - announce**

announceToDelegate

- **announceToDelegate;**

Calls the delegate's **errorWasAnnounced:** method, if it exists. (If it doesn't, nothing happens.) Returns **self** .

See also: **- delegate, - announce**

announceToStderr

- **announceToStderr;**

Prints the header string and complete message string to **stderr** . Returns **self** .

See also: **- headerString, - completeMessageString, - announce**

announceWithLogError

- **announceWithLogError;**

Writes the header and complete message string using NXLogError(), which sends it to the Console or **stderr** depending on whether the application was launched from the Workspace Manager or a shell. Returns **self** .

See also: **- headerString, - completeMessageString, - announce**

announceWithPanel

- **announceWithPanel;**

Runs an Alert Panel with the header string as the *title* and the complete message string as the *msg*. If the delegate responds to **provideHelpAbout:**, the *alternateButton* will be labelled "Help..."; pressing this button will perform the delegate's **provideHelpAbout:** method.

If running the alert panel fails for some reason, this method falls back on **announceByRaisingException** .

Returns **self** .

See also: **- headerString, - completeMessageString, - delegate, - announce**

announceWithSound

- **announceWithSound;**

Plays the system beep. Returns **self** .

See also: **- announce, NXBeep()**

announceWithSyslog

- **announceWithSyslog;**

Writes the header and complete message string onto the system log using the **syslog()** call, at the LOG_ERR priority. Returns **self**.

See also: **- headerString, - completeMessageString, - announce**

completeMessage

- **completeMessage;**

Returns the complete message string: the message string with parameters filled in by the **completeMessageString:** method. Do not free this FCString or modify it yourself, as it's owned by the FCError object.

See also: **- completeMessageString, - completeMessageString:, - message**

completeMessageString

- (const char *)**completeMessageString;**

Returns the complete message string: the message string with parameters filled in by the **completeMessageString:** method.

See also: - completeMessage, - completeMessageString:, - messageString

completeMessageString:...

- **completeMessageString:**(char *)*messageString,* **...;**

Sets the parameters for the FCError object's message string (which is a **printf()** format string).

See also: - setMessage:, - setMessageString:, - completeMessage, - completeMessageString

copyFromZone:

- **copyFromZone:**(NXZone *)*zone;*

Returns an unregistered duplicate of the FCError.

delegate

- **delegate;**

Returns the receiver's delegate.

See also: **- setDelegate:**

deregister

- **deregister;**

Removes receiver from the error registry. Returns **nil** if the error was not registered.

free

- **free;**

Frees the FCError. If the delegate responds to **errorDidFree:**, it is notified.

header

- **header;**

Returns the FCString containing the receiver's header. Do not free this FCString or modify it yourself, as it's owned by the FCError object.

See also: **- headerString, - setHeader:**

headerString

- (const char *)**headerString;**

Returns the FCError object's header string as a (char *).

See also: **- header, - setHeaderString:**

init

- **init;**

Initializes the FCError, which must be a newly allocated FCError instance. By default, the header string is set to "Error" and the message string is set to "Unknown error". Returns **self**.

initWithHeaderString:messageString:

- **initWithHeaderString:**(char *)*headerString*
  messageString:(char *)*messageString;*

Initializes the FCError, which must be a newly allocated FCError instance. Sets the header string and message string to be *headerString* and *messageString* respectively. This method is the designated initializer for the FCError class.

message

- **message;**

Returns the FCString containing the receiver's message. Do not free this FCString or modify it yourself, as it's owned by the FCError object.

See also: **- messageString, - setMessage:, - completeMessage**

messageString

- (const char *)**messageString;**

Returns the FCError object's message string as a (char *).

See also: **- message, - setMessageString:, - completeMessageString**

preferredAnnouncer

- (SEL)**preferredAnnouncer;**

Returns either the selector sent to the FCError object when the **announce** message is received, or FC_DEFAULT_ANNOUNCER.

See also: **- setPreferredAnnouncer:, + defaultAnnouncer, - announce**

read:

- **read:**(NXTypedStream *)*stream;*

Reads the receiver's instance variables from the typed stream *stream* .

This method is only used by the Objective-C archiving functions. You should not call this method directly.

See also: **- read:**

registerError:owner:

- **registerError:**(const char *)*theError* **owner:**(const char *)*theOwner;*

Enters receiver in the error registry under the name *theError* for the owner *theOwner*. Each owner has its own namespace.

registeredErrorString

- (const char *)**registeredErrorString;**

Returns the error string that was used to register the receiver. If the receiver is unregistered then this method returns NULL.

registeredOwnerString

- (const char *)**registeredOwnerString;**

Returns the owner string that was used to register the receiver. If the receiver is unregistered then this method returns NULL.

setDelegate:

- **setDelegate:anObject;**

Makes *anObject* the receiver's delegate.

See also: **- delegate**

setHeader:

- **setHeader:theHeader;**

Sets the FCError object's header string. The argument *theMessage* should be an FCString. Common examples of header strings are "Error", "Warning", and "Message".

See also: **- setHeaderString:, - header**

setHeaderString:

- **setHeaderString:**(const char *)*theHeader;*

Sets the receiver's header string. Common examples of header strings are "Error", "Warning", and "Message".

See also: **- setHeader:, - headerString**

setMessage:

- **setMessage:theMessage;**

Sets the receiver's message string. The argument *theMessage* should be an FCString. The message may be a **printf()** format string; parameters may be set by using **completeMessageString:**. This method works by invoking the **setMessageString:** method. Returns **self**.

See also: - setMessageString:, - completeMessageString:, - message

setMessageString:

- **setMessageString:**(const char *)*theMessage;*

Sets the FCError object's message string. The message may be a **printf()** format string; parameters may be set by using **completeMessageString:**. Returns **self**.

See also: - setMessage:, - completeMessageString:, - messageString

setPreferredAnnouncer:

- **setPreferredAnnouncer:**(SEL)*thePreferredAnnouncer;*

Sets the selector to be sent to the FCError object when the **announce** message is received. You may pass in FC_DEFAULT_ANNOUNCER to cause receiver to use the default FCError announcer selector.

See also: **- preferredAnnouncer, + setDefaultAnnouncer:, - announce**

write:

- **write:**(NXTypedStream *)*stream;*

Writes the receiver's instance variables to the typed stream *stream* .

This method is only used by the Objective-C archiving functions. You should not call this method directly.

See also: **- read:**

Methods Implemented by the Delegate

errorDidFree:

- **errorDidFree:anError;**

Sent to the delegate when *anError* is freed.

See also: **- free**

errorWasAnnounced:

- **errorWasAnnounced:anError;**

Sent to the delegate when *anError* announces an error using the **announceToDelegate** method.

See also: **- announceToDelegate**

provideHelpAbout:

- **provideHelpAbout:anError;**

If *anError's* delegate implements this method, the **announceWithPanel** method will include a button labelled "Help..." on its panel. If the user presses this button, the delegate will be notified via this message, and is expected to respond by providing the user with some help.

See also: **- announceWithPanel**

Macros

FC_ASSERTION()

FC_ASSERTION(*conditional* , *string* **)**

If *conditional* is FALSE, announce *string* using the pre-registered FCError *bugCustomAssertionFailed* ; or, if *string* is **NULL** , announce "Assertion failed: File **__FILE__** , line **__LINE__** " using the pre-registered FCError *bugDefaultAssertionFailed*. The method used to announce the message is the default announcer for **__FILE__**. If no announcer is assigned to **__FILE__** , the FCError's preferred

announcer will be used.

(**__FILE__** and **__LINE__** are compiler directives which will be replaced with the filename and line number in which they occur.)

See also: **FC_BASE_ASSERTION(), FC_PRECONDITION(), FC_POSTCONDITION(), FC_CLAIM(), - announcerForFile:, - announce, - preferredAnnouncer**

FC_CLAIM()

FC_CLAIM(*conditional* , *string* **)**

If *conditional* is FALSE, announce *string* using the pre-registered FCError *bugCustomAssertionFailed* ; or, if *string* is **NULL** , announce "Claim failed: File **__FILE__** , line **__LINE__** " using the pre-registered FCError *bugDefaultAssertionFailed*. The method used to announce the message is the default announcer for **__FILE__**. If no announcer is assigned to **__FILE__** , the FCError's preferred announcer will be used.

(**__FILE__** and **__LINE__** are compiler directives which will be replaced with the filename and line number in which they occur.)

The messages generated by this macro can be disabled by compiling with **-D** FC_BLOCKCLAIMS.

See also: **FC_BASE_ASSERTION(), FC_ASSERTION(), FC_PRECONDITION(), FC_POSTCONDITION(), - announcerForFile:, - announce, - preferredAnnouncer**

FC_EXCEPTION_RAISED()

FC_EXCEPTION_RAISED

Call this macro in the NX_HANDLER part of an exception handler to get the **id** of the FCError object that raised the exception. If the exception wasn't raised by an FCError object, **nil** is returned.

FC_POSTCONDITION()

FC_POSTCONDITION(*conditional* , *string* **)**

If *conditional* is FALSE, announce *string* using the pre-registered FCError *bugCustomAssertionFailed* ; or, if *string* is **NULL** , announce "Postcondition failed: File **__FILE__** , line **__LINE__** " using the pre-registered FCError *bugDefaultAssertionFailed*. The method used to announce the message is the default announcer for **__FILE__**. If no announcer is assigned to **__FILE__** , the FCError's preferred announcer will be used.

(**__FILE__** and **__LINE__** are compiler directives which will be replaced with the filename and line number in which they occur.)

See also: FC_BASE_ASSERTION(), FC_ASSERTION(), FC_PRECONDITION(), FC_CLAIM(), -announcerForFile:, - announce, - preferredAnnouncer

FC_PRECONDITION()

FC_PRECONDITION(*conditional* , *string* **)**

If *conditional* is FALSE, announce *string* using the pre-registered FCError *bugCustomAssertionFailed* ; or, if *string* is **NULL** , announce "Precondition failed: File **__FILE__** , line **__LINE__** " using the pre-registered FCError *bugDefaultAssertionFailed*. The method used to announce the message is the default announcer for **__FILE__**. If no announcer is assigned to **__FILE__** , the FCError's preferred announcer will be used.

(**__FILE__** and **__LINE__** are compiler directives which will be replaced with the filename and line number in which they occur.)

See also: FC_BASE_ASSERTION(), FC_ASSERTION(), FC_POSTCONDITION(), FC_CLAIM(), - announcerForFile:, - announce, - preferredAnnouncer

_FC_BASE_ASSERTION()

_FC_BASE_ASSERTION(*conditional* , *string* , *type* **)**

If *conditional* is FALSE, announce *string* using the pre-registered FCError *bugCustomAssertionFailed* ; or, if *string* is **NULL** , announce "*type* failed: File **__FILE__** , line **__LINE__** " using the pre-registered FCError *bugDefaultAssertionFailed* . The method used to announce the message is the default announcer for **__FILE__**. If no announcer is assigned to **__FILE_**, the FCError's preferred announcer will be used.

(**__FILE__** and **__LINE__** are compiler directives which will be replaced with the filename and line number in which they occur.)

See also: FC_ASSERTION(), FC_PRECONDITION(), FC_POSTCONDITION(), FC_CLAIM(), - announcerForFile:, - announce