

Copyright 1992 by Nik A Gervae. This is part of the documentation for the socket classes, which are licensed under the terms of the GNU General Public License as published by the Free Software Foundation.

The documented program and this documentation are distributed in the hope that it will be useful, but are provided "AS IS" AND WITHOUT ANY WARRANTY; without any express or implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. Any use or distribution of the program and documentation must include appropriate copyrights to acknowledge Nik A. Gervae and the Free Software Foundation, Inc.

You should have received a copy of the GNU General Public License along with this documentation; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

=====

SktsSocket

| | |
|---------------|--------------|
| INHERITS FROM | Object |
| DECLARED IN | SktsSocket.h |

CLASS DESCRIPTION

The SktsSocket class, alone, provides an application with a clean object interface to Berkeley UNIX stream socket connections, or together with the SktsSocketManager and SktsSocketUser classes, provides an application with the ability to act as a server for stream socket connections. The SktsSocket class implements the actual input and output on the socket (the distinction between SktsSocket and socket parallels that between Window and window). The class specifications for SktsSocketManager and SktsSocketUser provide more information on using these three classes together.

An SktsSocket object is automatically created and maintained by an

application's SktSocketManager object. If you use an SktSocketManager, you need never directly access the SktSockets.

These classes are intended to make sockets easy to use. However, several methods provide ways to access fairly low-level details of sockets which may be useful in some situations. For more information on sockets, see: the related UNIX man pages; ^aAn Introductory 4.3BSD Interprocess Communication Tutorial^o (reprinted in UNIX Programmer's Supplementary Documents Volume 1, PS1:7); or, ^aAn Advanced 4.3BSD Interprocess Communication Tutorial^o (reprinted in UNIX Programmer's Supplementary Documents Volume 1, PS1:8).

See also: SktSocketManager, SktSocketUser

INSTANCE VARIABLES

| | | |
|------------------------------|--|---------------|
| <i>Inherited from Object</i> | Class | isa; |
| <i>Declared in SktSocket</i> | int | socketFd; |
| | SktSocketManager | *manager; |
| | SktSocketUser | *user; |
| | char | *hostaddress; |
| | char | *hostname |
| | char | *outputQueue; |
| | long int | queueLength; |
| | NXZone | *zone; |
| SktSocketFd | The file descriptor of the SktSocket. | |
| manager | The SktSocket object's SktSocketManager. | |
| user | The SktSocket object's SktSocketUser. | |
| hostaddress | The Internet address of the SktSocket's connected host (peer) in dot notation. | |

| | |
|-------------|---|
| hostname | The hostname of the SktSocket's connected host. |
| outputQueue | Strings waiting to be sent to the connected host. |
| queueLength | The length, in bytes, of the output queue. |
| zone | The zone that the SktSocket was allocated from. |

METHOD TYPES

Initializing and freeing SktSocket objects

- €initOnFd:WithManager:
- €initOnHostname:andPort:
- €initOnAddress:andPort:
- €setSocketOptions:
- €close
- €free

Retrieving SktSocket attributes

- €setUser:
- €user
- €manager
- €socketFd
- €hostaddress
- €hostname

Managing input and output

- €readInput
- €queueOutput:ofLength:
- €queueOutputString:
- €flushOutput
- €purgeOutput

INSTANCE METHODS

close

- close

If the SktSocket has a manager, sends the SktSocketManager a **closeSocket:** message to properly close the socket and remove all references to the Socket object. This will result in **free** being sent to the Socket, so you should not send a **free** message after a **close**. If the SktSocket does not have a manager, this method simply sends **self** a **free** message. Returns **nil**.

See also: - **free**

flushOutput

-flushOutput

Writes all data in the output queue to the socket, thereby emptying the queue. Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocket has become corrupt, and will crash the process the next time it attempts to alter its output queue. You should either free the SktSocket or terminate the process.

See also: -**queueOutput:**, -**purgeOutput**

free

-free

Purges all pending output, closes the socket, and deallocates the SktSocket's memory. You should not send this message yourself; use **close** to correctly remove all references to the Socket in the associated SktSocketManager.

See also: -**close**, -**purgeOutput**

hostaddress

-€(const char *)hostaddress

Returns the Internet address of the SktSocket's connected host in dot notation.

See also: **-€hostname**

hostname

-€(const char *)hostname

Returns the hostname of the SktSocket's connect host.

See also: **-€hostaddress**

initOnAddress:andPort:

-€initOnAddress:(const char *)hostAddress andPort:(int)port

Attempts to connect to a server on the host at address *hostAddress* (a string in Internet dot notation), port *port*. The SktSocket's host address and name are recorded. If all goes well, returns **self**; otherwise it returns **nil**. Error messages are printed to stderr detailing the reason for failure.

This method also invokes **setSocketOptions:**, so that you may set file descriptor flags in subclasses. If **setSocketOptions:** returns **nil**, the initialization is cancelled and returns **nil**. SktSockets set the FNDELAY file descriptor flag for themselves (see *fcntl(2)* in the UNIX manual pages), so you needn't set that option.

This method, along with **initOnHostname:andPort:** and **initOnFd:withManager:**, are the designated initializers for SktSocket

object. Be sure to send one of these messages to **super** in any subclass **init...** methods based on your needs.

See also: **-[initOnHostname:andPort:]**, **-[initOnFd:withManager:]**, **-[setSocketOptions:]**, **-[setUser:]**

initOnFd:withManager:

-[init OnFd:(int)serviceSocketFd withManager:
(SktSocketManager*)aManager

Attempts to accept a connection to the sender's service socket with file descriptor *serviceSocketFd*. *aManager* must be of class *SktSocketManager* or a subclass. The *SktSocket*'s host address and name are recorded. If all goes well, returns **self**; otherwise it returns **nil**. If a *SktSocketManager* is specified, it will log any error messages.

This method also invokes **setSocketOptions:**, so that you may set file descriptor flags in subclasses. If **setSocketOptions:** returns **nil**, the initialization is cancelled and returns **nil**. *SktSockets* set the *FNDELAY* file descriptor flag for themselves (see *fcntl(2)* in the UNIX manual pages), so you needn't set that option.

This method, along with **initOnAddress:andPort:** and **initOnHostname:andPort:**, are the designated initializers for *SktSocket* object. Be sure to send one of these messages to **super** in any subclass **init...** methods based on your needs.

See also: **-[initOnAddress:andPort:]**, **-[initOnHostname:andPort:]**, **-[setSocketOptions:]**, **-[setUser:]**

initOnHostname:andPort:

-[initOnHostname:(const char *)hostName andPort:(int)port

Attempts to connect to a server on host *hostName* at port *port*. The

SktSocket's host address and name are recorded. If all goes well, returns **self**; otherwise it returns **nil**. Error messages are printed to stderr detailing the reason for failure.

This method also invokes **setSocketOptions:**, so that you may set file descriptor flags in subclasses. If **setSocketOptions:** returns **nil**, the initialization is cancelled and returns **nil**. SktSockets set the FNDELAY file descriptor flag for themselves (see *fcntl(2)* in the UNIX manual pages), so you needn't set that option.

This method, along with **initWithAddress:andPort:** and **initWithFd:withManager:**, are the designated initializers for SktSocket object. Be sure to send one of these messages to **super** in any subclass **init...** methods based on your needs.

See also: **-initWithAddress:andPort:**, **-initWithFd:withManager:**, **-setSocketOptions:**, **-setUser:**

manager

-initWithFd:withManager:manager

Returns the SktSocketManager object coordinating this SktSocket with others. This is a handy test to find out if an SktSocket is running as a server or a client.

See also: **-initWithFd:withManager:**

purgeOutput

-purgeOutput

Empties the output queue without writing it to the socket. Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocket has become corrupt, and will crash the process the next time it attempts to alter its output queue. You should either free the

SktSocket or terminate the process.

See also: **-€flushOutput**, **-€free**

queueOutput:ofLength:

-€queueOutput:(const char *)output ofLength:(long int)length

This method, usually invoked by the SktSocketUser, adds *length* bytes of *output* to the end of the output queue, and updates the queue length. Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocket has become corrupt, and will crash the process the next time it attempts to alter its output queue. You should either free the SktSocket or terminate the process.

See also: **-€flushOutput**, **-€purgeOutput**

queueOutputString:

-€queueOutputString:(const char *)aString

This method, usually invoked by the SktSocketUser, adds *aString* to the end of the output queue, and updates the queue length. It is provided as a convenience, and is equivalent to sending

```
[myUser queueOutput:aString ofLength:strlen(aString)];
```

It is therefore not useful for having the null character at the end of the string queued. To do that, add 1 to *aString*'s length and send **queueOutput:ofLength:.**

Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocket has become corrupt, and will crash the process the next time it attempts to alter its output queue. You should either free the SktSocket or terminate the process.

See also: **-€flushOutput**, **-€purgeOutput**

readInput

-readInput

This method, usually invoked by the SktSocket's manager, reads data from the socket and has the SktSocket's user queue the input, sending it a **queueInput:ofLength:** message. Returns **self**.

See also: **-queueInput:ofLength:** (SktSocketUser)

setSocketOptions:

-setSocketOptions:(int)fd

Does nothing and returns **self**. You can override this method in subclasses to set any options that you like on the socket's file descriptor. Be sure to send **setSocketOptions:** to **super** in your own method. If an operation fails so that you don't want the SktSocket to be initialized, this method should return **nil**. An SktSocket sets the FNDELAY flag for itself (see *fcntl(2)* in the UNIX manual pages), so you don't need to set that one.

See also: **-initWithFd:withManager:**, **-initWithAddress:andPort:**, **-initWithHostname:andPort:**, *fcntl(2)*, *setsockopt(2)*

setUser:

-setUser:aUser

Sets the SktSocket's user (which must be a subclass of SktSocketUser) to *aUser* and returns the old user. Also has the user set its socket to the SktSocket.

See also: **-initWith:manager:**

socketFd

-€(int)socketFd

Returns the file descriptor of the SktSocket's socket.

See also: **-€initWithManager:**

user

-€user

Returns the the SktSocket's SktSocketUser object.

See also: **-€setUser:**

CONSTANTS AND DEFINED TYPES

```
/*
 * Minimum size of the output queue. It gets shrunk
 * to this if the content of the queue gets
 * significantly smaller than this.
 */
#define OUTQSIZE 1024
```