SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [File List](http://docs.google.com/files.htm)
* [include](http://docs.google.com/dir_f3190241575fd2bd132a392ae6942f4a.htm)
* [SFML](http://docs.google.com/dir_692f376662c82a26cfe4cfa3aceebe24.htm)
* [Network](http://docs.google.com/dir_b9ac88db2949395b3130dd4ffb1be4e1.htm)

Socket.hpp

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2013 Laurent Gomila (laurent.gom@gmail.com)

5 //

6 // This software is provided 'as-is', without any express or implied warranty.

7 // In no event will the authors be held liable for any damages arising from the use of this software.

8 //

9 // Permission is granted to anyone to use this software for any purpose,

10 // including commercial applications, and to alter it and redistribute it freely,

11 // subject to the following restrictions:

12 //

13 // 1. The origin of this software must not be misrepresented;

14 // you must not claim that you wrote the original software.

15 // If you use this software in a product, an acknowledgment

16 // in the product documentation would be appreciated but is not required.

17 //

18 // 2. Altered source versions must be plainly marked as such,

19 // and must not be misrepresented as being the original software.

20 //

21 // 3. This notice may not be removed or altered from any source distribution.

22 //

24

25 #ifndef SFML\_SOCKET\_HPP

26 #define SFML\_SOCKET\_HPP

27

29 // Headers

31 #include <SFML/Network/Export.hpp>

32 #include <SFML/Network/SocketHandle.hpp>

33 #include <SFML/System/NonCopyable.hpp>

34 #include <vector>

35

36

37 namespace sf

38 {

39 class SocketSelector;

40

[45](http://docs.google.com/classsf_1_1Socket.htm) class SFML\_NETWORK\_API [Socket](http://docs.google.com/classsf_1_1Socket.htm) : [NonCopyable](http://docs.google.com/classsf_1_1NonCopyable.htm)

46 {

47 public :

48

[53](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dc)  enum [Status](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dc)

54  {

[55](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca1de3a85bc56d3ae85b3d0f3cfd04ae90)  [Done](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca1de3a85bc56d3ae85b3d0f3cfd04ae90),

[56](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca8554848daae98f996e131bdeed076c09)  [NotReady](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca8554848daae98f996e131bdeed076c09),

[57](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dcab215141f756acdc23c67fad149710eb1)  [Disconnected](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dcab215141f756acdc23c67fad149710eb1),

[58](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca1dc9854433a28c22e192721179a2df5d)  Error

59  };

60

65  enum

66  {

[67](http://docs.google.com/classsf_1_1Socket.htm#a5deb2c955fd347259c3a20d27b2481aaa5a3c30fd128895403afc11076f461b19)  AnyPort = 0

68  };

69

70 public :

71

76  virtual ~[Socket](http://docs.google.com/classsf_1_1Socket.htm)();

77

95  void setBlocking(bool blocking);

96

105  bool isBlocking() const;

106

107 protected :

108

[113](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8)  enum [Type](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8)

114  {

[115](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8acc02e97e90234b957eaad4dff7f22214)  [Tcp](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8acc02e97e90234b957eaad4dff7f22214),

[116](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8a6ebf3094830db4820191a327f3cc6ce2)  Udp

117  };

118

127  [Socket](http://docs.google.com/classsf_1_1Socket.htm)(Type type);

128

139  SocketHandle getHandle() const;

140

147  void create();

148

158  void create(SocketHandle handle);

159

166  void close();

167

168 private :

169

170  friend class [SocketSelector](http://docs.google.com/classsf_1_1SocketSelector.htm);

171

173  // Member data

175  Type m\_type;

176  SocketHandle m\_socket;

177  bool m\_isBlocking;

178 };

179

180 } // namespace sf

181

182

183 #endif // SFML\_SOCKET\_HPP

184

185

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::