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)

Packet.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\_PACKET\_HPP

26 #define SFML\_PACKET\_HPP

27

29 // Headers

31 #include <SFML/Network/Export.hpp>

32 #include <string>

33 #include <vector>

34

35

36 namespace sf

37 {

38 class String;

39 class TcpSocket;

40 class UdpSocket;

41

[47](http://docs.google.com/classsf_1_1Packet.htm) class SFML\_NETWORK\_API [Packet](http://docs.google.com/classsf_1_1Packet.htm)

48 {

49  // A bool-like type that cannot be converted to integer or pointer types

50  typedef bool ([Packet](http://docs.google.com/classsf_1_1Packet.htm)::\*BoolType)(std::size\_t);

51

52 public :

53

60  [Packet](http://docs.google.com/classsf_1_1Packet.htm)();

61

66  virtual ~[Packet](http://docs.google.com/classsf_1_1Packet.htm)();

67

77  void append(const void\* data, std::size\_t sizeInBytes);

78

87  void clear();

88

102  const void\* getData() const;

103

115  std::size\_t getDataSize() const;

116

129  bool endOfPacket() const;

130

131 public:

132

171  operator BoolType() const;

172

177  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(bool& data);

178  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(Int8& data);

179  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(Uint8& data);

180  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(Int16& data);

181  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(Uint16& data);

182  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(Int32& data);

183  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(Uint32& data);

184  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(float& data);

185  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(double& data);

186  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(char\* data);

187  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(std::string& data);

188  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(wchar\_t\* data);

189  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>(std::wstring& data);

190  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator >>([String](http://docs.google.com/classsf_1_1String.htm)& data);

191

196  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(bool data);

197  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(Int8 data);

198  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(Uint8 data);

199  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(Int16 data);

200  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(Uint16 data);

201  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(Int32 data);

202  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(Uint32 data);

203  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(float data);

204  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(double data);

205  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(const char\* data);

206  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(const std::string& data);

207  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(const wchar\_t\* data);

208  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(const std::wstring& data);

209  [Packet](http://docs.google.com/classsf_1_1Packet.htm)& operator <<(const [String](http://docs.google.com/classsf_1_1String.htm)& data);

210

211 protected:

212

213  friend class [TcpSocket](http://docs.google.com/classsf_1_1TcpSocket.htm);

214  friend class [UdpSocket](http://docs.google.com/classsf_1_1UdpSocket.htm);

215

234  virtual const void\* onSend(std::size\_t& size);

235

253  virtual void onReceive(const void\* data, std::size\_t size);

254

255 private :

256

261  bool operator ==(const [Packet](http://docs.google.com/classsf_1_1Packet.htm)& right) const;

262  bool operator !=(const [Packet](http://docs.google.com/classsf_1_1Packet.htm)& right) const;

263

274  bool checkSize(std::size\_t size);

275

277  // Member data

279  std::vector<char> m\_data;

280  std::size\_t m\_readPos;

281  bool m\_isValid;

282 };

283

284 } // namespace sf

285

286

287 #endif // SFML\_PACKET\_HPP

288

289

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