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)

Ftp.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\_FTP\_HPP

26 #define SFML\_FTP\_HPP

27

29 // Headers

31 #include <SFML/Network/Export.hpp>

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

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

34 #include <SFML/System/Time.hpp>

35 #include <string>

36 #include <vector>

37

38

39 namespace sf

40 {

41 class IpAddress;

42

[47](http://docs.google.com/classsf_1_1Ftp.htm) class SFML\_NETWORK\_API [Ftp](http://docs.google.com/classsf_1_1Ftp.htm) : [NonCopyable](http://docs.google.com/classsf_1_1NonCopyable.htm)

48 {

49 public :

50

[55](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cb)  enum [TransferMode](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cb)

56  {

[57](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cba6f253b362639fb5e059dc292762a21ee)  [Binary](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cba6f253b362639fb5e059dc292762a21ee),

[58](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cbac9e544a22dce8ef3177449cb235d15c2)  [Ascii](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cbac9e544a22dce8ef3177449cb235d15c2),

[59](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cbabb1e34435231e73c96534c71090be7f4)  Ebcdic

60  };

61

[66](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm)  class SFML\_NETWORK\_API [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm)

67  {

68  public :

69

[74](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3b)  enum [Status](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3b)

75  {

76  // 1xx: the requested action is being initiated,

77  // expect another reply before proceeding with a new command

[78](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba07e06d3326ba2d078583bef93930d909)  RestartMarkerReply = 110,

[79](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba22413357ade6b586f6ceb0d704f35075)  ServiceReadySoon = 120,

[80](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bafa52d19bc813d69055f4cc390d4a76ca)  DataConnectionAlreadyOpened = 125,

[81](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba794ebe743688be611447638bf9e49d86)  OpeningDataConnection = 150,

82

83  // 2xx: the requested action has been successfully completed

[84](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baa956e229ba6c0cdf0d88b0e05b286210)  Ok = 200,

[85](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba38adc424f1adcd332745de8cd3b7737a)  PointlessCommand = 202,

[86](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba9bdd02ae119b8be639e778859ee74060)  SystemStatus = 211,

[87](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba8729460a695013cc96330e2fced0ae1f)  DirectoryStatus = 212,

[88](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baebddfc7997dca289c83068dff3f47dce)  FileStatus = 213,

[89](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba840fd2a1872fd4310b046541f57fdeb7)  HelpMessage = 214,

[90](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba78391f73aa11f07f1514c7d070b93c08)  SystemType = 215,

[91](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baea2ee2007d7843c21108bb686ef03757)  ServiceReady = 220,

[92](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bab23931490fc2d1df3081d651fe0f4d6e)  ClosingConnection = 221,

[93](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3badc78ed87d5bddb174fa3c16707ac2f2d)  DataConnectionOpened = 225,

[94](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bac723ebc8a38913bbf0d9504556cbaaa6)  ClosingDataConnection = 226,

[95](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba48314fc47a72ad0aacdea93b91756f6e)  EnteringPassiveMode = 227,

[96](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba54a88210386cb72e35d737813a221754)  LoggedIn = 230,

[97](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baf988b69b0a5f55f8122da5ba001932e0)  FileActionOk = 250,

[98](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba06d26e95a170fc422af13def415e0437)  DirectoryOk = 257,

99

100  // 3xx: the command has been accepted, but the requested action

101  // is dormant, pending receipt of further information

[102](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba9249e3fe9818eb93f181fbbf3ae3bc56)  NeedPassword = 331,

[103](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba9e048185f253f6eb6f5ff9e063b712fa)  NeedAccountToLogIn = 332,

[104](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba02e6f05964ecb829e9b6fb6020d6528a)  NeedInformation = 350,

105

106  // 4xx: the command was not accepted and the requested action did not take place,

107  // but the error condition is temporary and the action may be requested again

[108](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba43022ddf49b68a4f5aff0bea7e09e89f)  ServiceUnavailable = 421,

[109](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba757b89ff1f236941f7759b0ed0c28b88)  DataConnectionUnavailable = 425,

[110](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba7cfefcc586c12ba70f752353fde7126e)  TransferAborted = 426,

[111](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baf822d1b0abf3e9ae7dd44684549d512d)  FileActionAborted = 450,

[112](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bae54e84baaca95a7b36271ca3f3fdb900)  LocalError = 451,

[113](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba5d9f3666222c808553c27e4e099c7c6d)  InsufficientStorageSpace = 452,

114

115  // 5xx: the command was not accepted and

116  // the requested action did not take place

[117](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba75bdf0b6844fa9c07b3c25647d22c269)  CommandUnknown = 500,

[118](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baf4c7c88815981bbb7c3a3461f9f48b67)  ParametersUnknown = 501,

[119](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba2ca4834c756c81b924ebed696fcba0a8)  CommandNotImplemented = 502,

[120](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bad0c7ab07f01c1f7af16a1852650d7c47)  BadCommandSequence = 503,

[121](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba8807473b8590e1debfb3740b7a3d081c)  ParameterNotImplemented = 504,

[122](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bafcfbaff2c6fed941b6bcbc0999db764e)  NotLoggedIn = 530,

[123](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba1af0f173062a471739b50d8e0f40d5f7)  NeedAccountToStore = 532,

[124](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba3f8f931e499936fde6b750d81f5ecfef)  FileUnavailable = 550,

[125](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bad220bc12dc45593af6e5079ea6c532c3)  PageTypeUnknown = 551,

[126](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baf418e54753e0b8f9cb0325dd618acd14)  NotEnoughMemory = 552,

[127](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba03254aba823298179a98056e15568c5b)  FilenameNotAllowed = 553,

128

129  // 10xx: SFML custom codes

[130](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba59e041e4ef186e8ae8d6035973fc46bd)  InvalidResponse = 1000,

[131](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3ba51aa367cc1e85a45ea3c7be48730e990)  ConnectionFailed = 1001,

[132](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3bad1e5dcf298ce30c528261435f1a2eb53)  ConnectionClosed = 1002,

[133](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm#af81738f06b6f571761696291276acb3baed2c74a9f335dee1463ca1a4f41c6478)  InvalidFile = 1003

134  };

135

146  explicit [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm)(Status code = InvalidResponse, const std::string& message = "");

147

157  bool isOk() const;

158

165  Status getStatus() const;

166

173  const std::string& getMessage() const;

174

175  private :

176

178  // Member data

180  Status m\_status;

181  std::string m\_message;

182  };

183

[188](http://docs.google.com/classsf_1_1Ftp_1_1DirectoryResponse.htm)  class SFML\_NETWORK\_API [DirectoryResponse](http://docs.google.com/classsf_1_1Ftp_1_1DirectoryResponse.htm) : public [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm)

189  {

190  public :

191

198  [DirectoryResponse](http://docs.google.com/classsf_1_1Ftp_1_1DirectoryResponse.htm)(const [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm)& response);

199

206  const std::string& getDirectory() const;

207

208  private :

209

211  // Member data

213  std::string m\_directory;

214  };

215

216

[221](http://docs.google.com/classsf_1_1Ftp_1_1ListingResponse.htm)  class SFML\_NETWORK\_API [ListingResponse](http://docs.google.com/classsf_1_1Ftp_1_1ListingResponse.htm) : public [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm)

222  {

223  public :

224

232  [ListingResponse](http://docs.google.com/classsf_1_1Ftp_1_1ListingResponse.htm)(const [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm)& response, const std::vector<char>& data);

233

240  const std::vector<std::string>& getListing() const;

241

242  private :

243

245  // Member data

247  std::vector<std::string> m\_listing;

248  };

249

250

258  ~[Ftp](http://docs.google.com/classsf_1_1Ftp.htm)();

259

281  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) connect(const [IpAddress](http://docs.google.com/classsf_1_1IpAddress.htm)& server, unsigned short port = 21, [Time](http://docs.google.com/classsf_1_1Time.htm) timeout = [Time::Zero](http://docs.google.com/classsf_1_1Time.htm#a8db127b632fa8da21550e7282af11fa0));

282

291  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) disconnect();

292

302  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) login();

303

316  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) login(const std::string& name, const std::string& password);

317

327  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) keepAlive();

328

340  [DirectoryResponse](http://docs.google.com/classsf_1_1Ftp_1_1DirectoryResponse.htm) getWorkingDirectory();

341

357  [ListingResponse](http://docs.google.com/classsf_1_1Ftp_1_1ListingResponse.htm) getDirectoryListing(const std::string& directory = "");

358

371  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) changeDirectory(const std::string& directory);

372

381  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) parentDirectory();

382

396  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) createDirectory(const std::string& name);

397

413  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) deleteDirectory(const std::string& name);

414

429  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) renameFile(const std::string& file, const std::string& newName);

430

446  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) deleteFile(const std::string& name);

447

465  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) download(const std::string& remoteFile, const std::string& localPath, [TransferMode](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cb) mode = Binary);

466

484  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) upload(const std::string& localFile, const std::string& remotePath, [TransferMode](http://docs.google.com/classsf_1_1Ftp.htm#a1cd6b89ad23253f6d97e6d4ca4d558cb) mode = Binary);

485

486 private :

487

497  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) sendCommand(const std::string& command, const std::string& parameter = "");

498

508  [Response](http://docs.google.com/classsf_1_1Ftp_1_1Response.htm) getResponse();

509

515  class DataChannel;

516

517  friend class DataChannel;

518

520  // Member data

522  [TcpSocket](http://docs.google.com/classsf_1_1TcpSocket.htm) m\_commandSocket;

523 };

524

525 } // namespace sf

526

527

528 #endif // SFML\_FTP\_HPP

529

530

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