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)
* [Class List](http://docs.google.com/annotated.htm)
* [Class Index](http://docs.google.com/classes.htm)
* [Class Hierarchy](http://docs.google.com/hierarchy.htm)
* [Class Members](http://docs.google.com/functions.htm)
* **sf**
* [Packet](http://docs.google.com/classsf_1_1Packet.htm)

[Public Member Functions](#_gjdgxs) | [Protected Member Functions](#_30j0zll) | [Friends](#_1fob9te) | [List of all members](http://docs.google.com/classsf_1_1Packet-members.htm)

sf::Packet Class Reference

[Network module](http://docs.google.com/group__network.htm)

Utility class to build blocks of data to transfer over the network. [More...](http://docs.google.com/classsf_1_1Packet.htm#details)

#include <[Packet.hpp](http://docs.google.com/Packet_8hpp_source.htm)>

| Public Member Functions | |
| --- | --- |
|  | [Packet](http://docs.google.com/classsf_1_1Packet.htm#a786e5d4ced83992ceefa1799963ea858) () |
|  | Default constructor. |
|  | |
| virtual | [~Packet](http://docs.google.com/classsf_1_1Packet.htm#adc0490ca3c7c3d1e321bd742e5213913) () |
|  | Virtual destructor. |
|  | |
| void | [append](http://docs.google.com/classsf_1_1Packet.htm#a7dd6e429b87520008326c4d71f1cf011) (const void \*data, std::size\_t sizeInBytes) |
|  | Append data to the end of the packet. |
|  | |
| void | [clear](http://docs.google.com/classsf_1_1Packet.htm#a133ea8b8fe6e93c230f0d79f19a3bf0d) () |
|  | Clear the packet. |
|  | |
| const void \* | [getData](http://docs.google.com/classsf_1_1Packet.htm#a304ba9ec94c992710f4dfff879c6340e) () const |
|  | Get a pointer to the data contained in the packet. |
|  | |
| std::size\_t | [getDataSize](http://docs.google.com/classsf_1_1Packet.htm#a004b62aa5bafa69df8917171a3fe1fa0) () const |
|  | Get the size of the data contained in the packet. |
|  | |
| bool | [endOfPacket](http://docs.google.com/classsf_1_1Packet.htm#aee3adfca6303f1e6bde3c62be392b945) () const |
|  | Tell if the reading position has reached the end of the packet. |
|  | |
|  | [operator BoolType](http://docs.google.com/classsf_1_1Packet.htm#addcb990cde37859c748273d9de55e628) () const |
|  | Test the validity of the packet, for reading. |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | [operator>>](http://docs.google.com/classsf_1_1Packet.htm#af8e26c63ba9bdccd262565ff0d3eeba2) (bool &data) |
|  | Overloads of operator >> to read data from the packet. |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (Int8 &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (Uint8 &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (Int16 &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (Uint16 &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (Int32 &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (Uint32 &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (float &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (double &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (char \*data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (std::string &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (wchar\_t \*data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** (std::wstring &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator>>** ([String](http://docs.google.com/classsf_1_1String.htm) &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | [operator<<](http://docs.google.com/classsf_1_1Packet.htm#aa5a465ed02ba29d83ecdafb0ac3fff21) (bool data) |
|  | Overloads of operator << to write data into the packet. |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (Int8 data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (Uint8 data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (Int16 data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (Uint16 data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (Int32 data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (Uint32 data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (float data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (double data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (const char \*data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (const std::string &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (const wchar\_t \*data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (const std::wstring &data) |
|  | |
| [Packet](http://docs.google.com/classsf_1_1Packet.htm) & | **operator<<** (const [String](http://docs.google.com/classsf_1_1String.htm) &data) |
|  | |

| Protected Member Functions | |
| --- | --- |
| virtual const void \* | [onSend](http://docs.google.com/classsf_1_1Packet.htm#a052e955906c9bfd671622cb625380edc) (std::size\_t &size) |
|  | Called before the packet is sent over the network. |
|  | |
| virtual void | [onReceive](http://docs.google.com/classsf_1_1Packet.htm#ab71a31ef0f1d5d856de6f9fc75434128) (const void \*data, std::size\_t size) |
|  | Called after the packet is received over the network. |
|  | |

| Friends | |
| --- | --- |
| class | **TcpSocket** |
|  | |
| class | **UdpSocket** |
|  | |

## Detailed Description

Utility class to build blocks of data to transfer over the network.

Packets provide a safe and easy way to serialize data, in order to send it over the network using sockets ([sf::TcpSocket](http://docs.google.com/classsf_1_1TcpSocket.htm), [sf::UdpSocket](http://docs.google.com/classsf_1_1UdpSocket.htm)).

Packets solve 2 fundamental problems that arise when transfering data over the network:

* data is interpreted correctly according to the endianness
* the bounds of the packet are preserved (one send == one receive)

The [sf::Packet](http://docs.google.com/classsf_1_1Packet.htm) class provides both input and output modes. It is designed to follow the behaviour of standard C++ streams, using operators >> and << to extract and insert data.

It is recommended to use only fixed-size types (like sf::Int32, etc.), to avoid possible differences between the sender and the receiver. Indeed, the native C++ types may have different sizes on two platforms and your data may be corrupted if that happens.

Usage example:

sf::Uint32 x = 24;

std::string s = "hello";

double d = 5.89;

// Group the variables to send into a packet

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

packet << x << s << d;

// Send it over the network (socket is a valid sf::TcpSocket)

socket.send(packet);

-----------------------------------------------------------------

// Receive the packet at the other end

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

socket.receive(packet);

// Extract the variables contained in the packet

sf::Uint32 x;

std::string s;

double d;

if (packet >> x >> s >> d)

{

// Data extracted successfully...

}

Packets have built-in operator >> and << overloads for standard types:

* bool
* fixed-size integer types (sf::Int8/16/32, sf::Uint8/16/32)
* floating point numbers (float, double)
* string types (char\*, wchar\_t\*, std::string, std::wstring, [sf::String](http://docs.google.com/classsf_1_1String.htm))

Like standard streams, it is also possible to define your own overloads of operators >> and << in order to handle your custom types.

struct MyStruct

{

float number;

sf::Int8 integer;

std::string str;

};

[sf::Packet](http://docs.google.com/classsf_1_1Packet.htm)& [operator <<](http://docs.google.com/classsf_1_1Packet.htm#aa5a465ed02ba29d83ecdafb0ac3fff21)([sf::Packet](http://docs.google.com/classsf_1_1Packet.htm)& packet, const MyStruct& m)

{

return packet << m.number << m.integer << m.str;

}

[sf::Packet](http://docs.google.com/classsf_1_1Packet.htm)& [operator >>](http://docs.google.com/classsf_1_1Packet.htm#af8e26c63ba9bdccd262565ff0d3eeba2)([sf::Packet](http://docs.google.com/classsf_1_1Packet.htm)& packet, MyStruct& m)

{

return packet >> m.number >> m.integer >> m.str;

}

Packets also provide an extra feature that allows to apply custom transformations to the data before it is sent, and after it is received. This is typically used to handle automatic compression or encryption of the data. This is achieved by inheriting from [sf::Packet](http://docs.google.com/classsf_1_1Packet.htm), and overriding the onSend and onReceive functions.

Here is an example:

class ZipPacket : public [sf::Packet](http://docs.google.com/classsf_1_1Packet.htm)

{

virtual const void\* [onSend](http://docs.google.com/classsf_1_1Packet.htm#a052e955906c9bfd671622cb625380edc)(std::size\_t& size)

{

const void\* srcData = [getData](http://docs.google.com/classsf_1_1Packet.htm#a304ba9ec94c992710f4dfff879c6340e)();

std::size\_t srcSize = [getDataSize](http://docs.google.com/classsf_1_1Packet.htm#a004b62aa5bafa69df8917171a3fe1fa0)();

return MySuperZipFunction(srcData, srcSize, &size);

}

virtual void [onReceive](http://docs.google.com/classsf_1_1Packet.htm#ab71a31ef0f1d5d856de6f9fc75434128)(const void\* data, std::size\_t size)

{

std::size\_t dstSize;

const void\* dstData = MySuperUnzipFunction(data, size, &dstSize);

[append](http://docs.google.com/classsf_1_1Packet.htm#a7dd6e429b87520008326c4d71f1cf011)(dstData, dstSize);

}

};

// Use like regular packets:

ZipPacket packet;

packet << x << s << d;

...

See Also[sf::TcpSocket](http://docs.google.com/classsf_1_1TcpSocket.htm), [sf::UdpSocket](http://docs.google.com/classsf_1_1UdpSocket.htm)

Definition at line [47](http://docs.google.com/Packet_8hpp_source.htm#l00047) of file [Packet.hpp](http://docs.google.com/Packet_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Packet::Packet | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

Creates an empty packet.

| | virtual sf::Packet::~Packet | ( |  | ) |  | | --- | --- | --- | --- | --- | | virtual |
| --- | --- | --- | --- | --- | --- | --- |

Virtual destructor.

## Member Function Documentation

| void sf::Packet::append | ( | const void \* | *data*, |
| --- | --- | --- | --- |
|  |  | std::size\_t | *sizeInBytes* |
|  | ) |  |  |

Append data to the end of the packet.

Parameters

| data | Pointer to the sequence of bytes to append |
| --- | --- |
| sizeInBytes | Number of bytes to append |

See Also[clear](http://docs.google.com/classsf_1_1Packet.htm#a133ea8b8fe6e93c230f0d79f19a3bf0d)

| void sf::Packet::clear | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Clear the packet.

After calling Clear, the packet is empty.

See Also[append](http://docs.google.com/classsf_1_1Packet.htm#a7dd6e429b87520008326c4d71f1cf011)

| bool sf::Packet::endOfPacket | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Tell if the reading position has reached the end of the packet.

This function is useful to know if there is some data left to be read, without actually reading it.

ReturnsTrue if all data was read, false otherwise See Alsooperator bool

| const void\* sf::Packet::getData | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get a pointer to the data contained in the packet.

Warning: the returned pointer may become invalid after you append data to the packet, therefore it should never be stored. The return pointer is NULL if the packet is empty.

ReturnsPointer to the data See Also[getDataSize](http://docs.google.com/classsf_1_1Packet.htm#a004b62aa5bafa69df8917171a3fe1fa0)

| std::size\_t sf::Packet::getDataSize | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the size of the data contained in the packet.

This function returns the number of bytes pointed to by what getData returns.

ReturnsData size, in bytes See Also[getData](http://docs.google.com/classsf_1_1Packet.htm#a304ba9ec94c992710f4dfff879c6340e)

| | virtual void sf::Packet::onReceive | ( | const void \* | *data*, | | --- | --- | --- | --- | |  |  | std::size\_t | *size* | |  | ) |  |  | | protectedvirtual |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Called after the packet is received over the network.

This function can be defined by derived classes to transform the data after it is received; this can be used for uncompression, decryption, etc. The function receives a pointer to the received data, and must fill the packet with the transformed bytes. The default implementation fills the packet directly without transforming the data.

Parameters

| data | Pointer to the received bytes |
| --- | --- |
| size | Number of bytes |

See Also[onSend](http://docs.google.com/classsf_1_1Packet.htm#a052e955906c9bfd671622cb625380edc)

| | virtual const void\* sf::Packet::onSend | ( | std::size\_t & | *size* | ) |  | | --- | --- | --- | --- | --- | --- | | protectedvirtual |
| --- | --- | --- | --- | --- | --- | --- | --- |

Called before the packet is sent over the network.

This function can be defined by derived classes to transform the data before it is sent; this can be used for compression, encryption, etc. The function must return a pointer to the modified data, as well as the number of bytes pointed. The default implementation provides the packet's data without transforming it.

Parameters

| size | Variable to fill with the size of data to send |
| --- | --- |

ReturnsPointer to the array of bytes to send See Also[onReceive](http://docs.google.com/classsf_1_1Packet.htm#ab71a31ef0f1d5d856de6f9fc75434128)

| sf::Packet::operator BoolType | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Test the validity of the packet, for reading.

This operator allows to test the packet as a boolean variable, to check if a reading operation was successful.

A packet will be in an invalid state if it has no more data to read.

This behaviour is the same as standard C++ streams.

Usage example:

float x;

packet >> x;

if (packet)

{

// ok, x was extracted successfully

}

// -- or --

float x;

if (packet >> x)

{

// ok, x was extracted successfully

}

Don't focus on the return type, it's equivalent to bool but it disallows unwanted implicit conversions to integer or pointer types.

ReturnsTrue if last data extraction from packet was successful See Also[endOfPacket](http://docs.google.com/classsf_1_1Packet.htm#aee3adfca6303f1e6bde3c62be392b945)

| [Packet](http://docs.google.com/classsf_1_1Packet.htm)& sf::Packet::operator<< | ( | bool | *data* | ) |  |
| --- | --- | --- | --- | --- | --- |

Overloads of operator << to write data into the packet.

| [Packet](http://docs.google.com/classsf_1_1Packet.htm)& sf::Packet::operator>> | ( | bool & | *data* | ) |  |
| --- | --- | --- | --- | --- | --- |

Overloads of operator >> to read data from the packet.

The documentation for this class was generated from the following file:

* [Packet.hpp](http://docs.google.com/Packet_8hpp_source.htm)

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