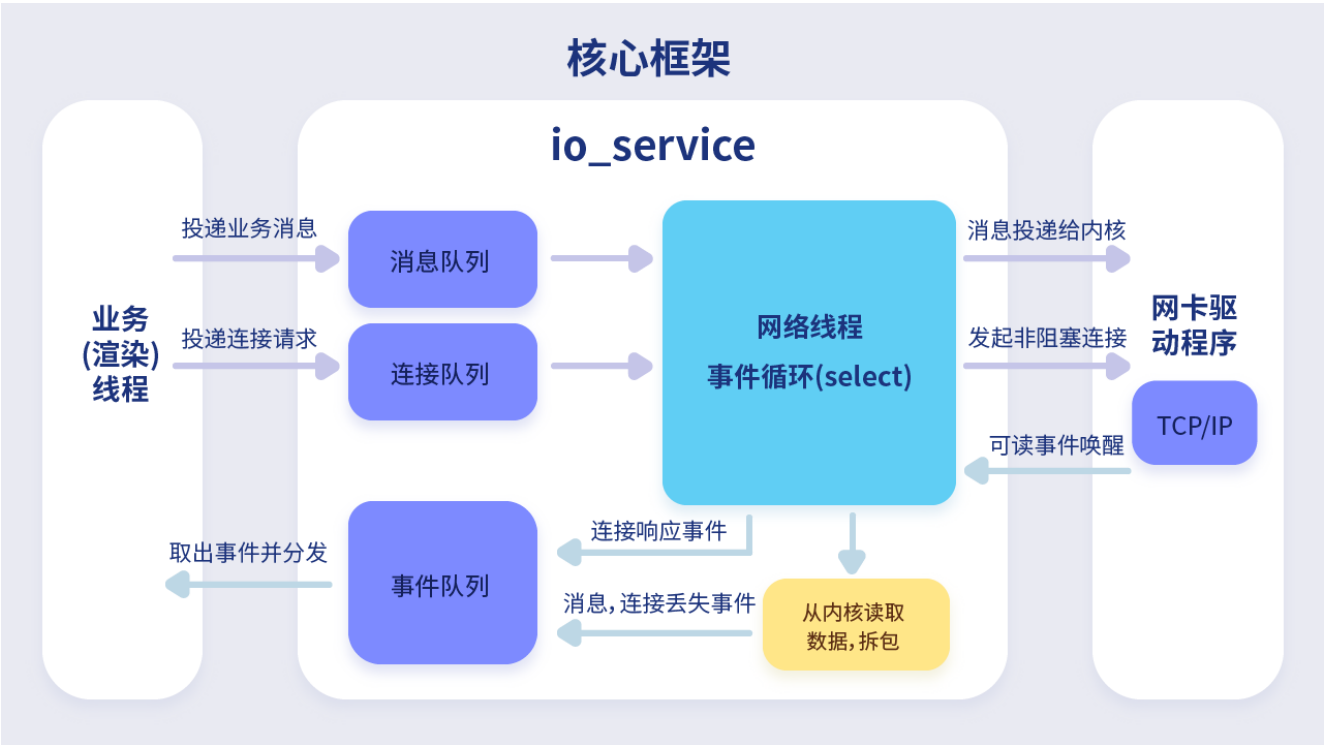


yasio 4F656568
2D878763

docs passing

yasio 2F4F2A7B217F2855F07265socket556C15E04E59377A54FA8E54C85B625E64766B625B377A517E6751B9

- 2855F327:
 - 7E8958:
 - Visual Studio 2013+
 - GCC4.7+
 - xcode9+
 - 5A6B2263 C++11,14,17 697E8928
 - 6764: x86, x64, ARM4B32
 - 644E797E: Windows, macOS, Linux, FreeBSD, iOS, Android7939
- 27E8:
 - 517E6D51: io_service
 - 4F29: io_channel
 - 4E851A88: io_transport
- 8862FE:



5F905F59
EB1F00CB

69584E7A8F68E3E4 tool.chinaz.com 6389http89655E537874651706289

C++

```
#include "yasio/yasio.hpp"
#include "yasio/obstream.hpp"
using namespace yasio;
using namespace yasio::inet;
int main()
{
    io_service service({"tool.chinaz.com", 80});
    service.set_option(YOPT_S_DEFERRED_EVENT, 0); // dispatch network event on network thread
    service.start([&](event_ptr&& ev) {
        switch (ev->kind())
        {
        case YEK_PACKET: {
            auto packet = std::move(ev->packet());
            fwrite(packet.data(), packet.size(), 1, stdout);
            fflush(stdout);
            break;
        }
        case YEK_CONNECT_RESPONSE:
            if (ev->status() == 0)
            {
                auto transport = ev->transport();
                if (ev->cindex() == 0)
                {
                    obstream obs;
                    obs.write_bytes("GET /index.htm HTTP/1.1\r\n");

                    obs.write_bytes("Host: tool.chinaz.com\r\n");

                    obs.write_bytes("User-Agent: Mozilla/5.0 (Windows NT 10.0; "
                                   "WOW64) AppleWebKit/537.36 (KHTML, like Gecko) "
                                   "Chrome/87.0.4820.88 Safari/537.36\r\n");
                    obs.write_bytes("Accept: */*;q=0.8\r\n");
                    obs.write_bytes("Connection: Close\r\n\r\n");

                    service.write(transport, std::move(obs.buffer()));
                }
            }
            break;
        case YEK_CONNECTION_LOST:
            printf("The connection is lost.\n");
            break;
        }
    });
    // open channel 0 as tcp client
    service.open(0, YCK_TCP_CLIENT);
    getchar();
}
```

Lua

```
local ip138 = "tool.chinaz.com"
local service = yasio.io_service.new({host=ip138, port=80})
local respdata = ""
```

```
service:start(function(ev)
    local k = ev.kind()
    if (k == yasio.YEK_PACKET) then
        respdata = respdata .. ev:packet():to_string()
    elseif k == yasio.YEK_CONNECT_RESPONSE then
        if ev:status() == 0 then -- connect succeed
            local transport = ev:transport()
            local obs = yasio.obstream.new()
            obs:write_bytes("GET / HTTP/1.1\r\n")

            obs:write_bytes("Host: " .. ip138 .. "\r\n")

            obs:write_bytes("User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36\r\n")
            obs:write_bytes("Accept: */*;q=0.8\r\n")
            obs:write_bytes("Connection: Close\r\n\r\n")

            service:write(transport, obs)
        end
    elseif k == yasio.YEK_CONNECTION_LOST then
        print("request finish, respdata: " .. respdata)
    end
end)

-- Open channel 0 as tcp client and start non-blocking tcp 3 times handshake
service:open(0, yasio.YCK_TCP_CLIENT)

-- Call this function at thread which focus on the network event.
function gDispatchNetworkEvent(...)
    service:dispatch(128) -- dispatch max events is 128 per frame
end

_G.yaservice = service -- Store service to global table as a singleton instance
```

6D8B & 794F
4B05 3A8B



8F88 Lua 794F7A5E FF4F62535F5E4F69 yasio - ibstream_view::consume out of range! FF8F69794F7A5E9169615176FF8B4E5F576139
6642 3A8B6B8F6C1A537662C8E16E 6C692F3A8B6B8F6C456F69846CF76D65286F62

- 6D8B:
 - echo_server: TCP/UDP/KCP 6B646D2168
 - echo_client: TCP/UDP/KCP 6B64A9627E
 - ssltest: SSL 6B89A9627E, 6969github.com 4B98766376846E626E
 - tcptest: TCP 4B897A6F

- [speedtest](#): TCP,UDP,KCP
- [mcast](#):
- [794F3A8B](#):
- [ftp_server](#): yasio ftp
- [lua](#): Lua http TCP
- [xlua](#): xlua
- [DemoUE4](#): UE4

7F8B 6D8B & 794F
16D1 4BD5 3A8B

- C++11 msvc , gcc , clang
- git , cmake installed
- 88485946704E:

```
git clone https://github.com/yasio/yasio
cd yasio
git submodule update --init --recursive
cd build
# for xcode should be: cmake .. -GXcode
cmake ..
cmake --build . --config Debug
```


io_service Class

```
socket.select tcp, udp, kcp, ssl-client
```

```
EDD5
```

```
namespace yasio { namespace inet { class io_service; } }
```

```
6254  
1058
```

```
515167905165  
6C718420FD70
```

Name	Description
io_service::io_service	io_service

```
6C516985
```

Name	Description
io_service::start	
io_service::stop	
io_service::open	
io_service::close	
io_service::is_open	
io_service::dispatch	
io_service::write	
io_service::write_to	DGRAM

Name	Description
io_service::schedule	
io_service::init_globals	
io_service::cleanup_globals	
io_service::channel_at	
io_service::set_option	

object_pool

yasio.hpp

io_service::io_service

io_service

```
io_service::io_service();

io_service::io_service(int channel_count);

io_service::io_service(const io_hostent& channel_ep);

io_service::io_service(const io_hostent* channel_eps, int channel_count);
```

channel_count

channel_ep

channel_eps

4F698F2A5757657499575789

794F
3A8B

```
#include "yasio/yasio.hpp"
int main() {
    using namespace yasio;
    using namespace yasio::inet;
    io_service s1; // s1 only support 1 channel
    io_service s2(5); // s2 support 5 channels concurrency
    io_service s3(io_hostent{"github.com", 443}); // s3 support 1 channel
    io_hostent hosts[] = {
        {"192.168.1.66", 20336},
        {"192.168.1.88", 20337},
    };
    io_service s4(hosts, YASIO_ARRAYSIZE(hosts)); // s4 support 2 channels concurrency
    return 0;
}
```

io_service::start

54527F7C67577F7A82

```
void start(io_event_cb_t cb);
```

5365
2278

cb

7F7E464E6983FF68A9C9534E57 io_service::dispatch 8975897E7A895E82

794F
3A8B

```
#include "yasio/yasio.hpp"
int main() {
    using namespace yasio;
    using namespace yasio::inet;
    auto service = yasio_shared_service(io_hostent{host="ip138.com", port=80});
    service->start([](event_ptr&& ev) {
        auto kind = ev->kind();
        if (kind == YEK_CONNECT_RESPONSE)
        {
            if (ev->status() == 0)
                printf("[%d] connect succeed.\n", ev->cindex());
            else
                printf("[%d] connect failed!\n", ev->cindex());
        }
    });
}
```

```
    return 0;
}
```

io_service::stop

59697f7e67527f7a39

```
void stop()
```

6061
e80f

59697f7e67527f7a69528f88, 6951654f539990514f535e795f51685e995139
64fd791a616160fae1f77649857663880fa62

io_service::open

525f4f9939

```
void open(size_t cindex, int kind);
```

5365
c270

cindex

4f53295f39

kind

4f53798939

6061
e80f

5948 TCP, 664af9425168529997985932f14b6267fa7aef978ef538
5948 TCP, 664af9425168529997985932f14b6267fa7aef978ef538

cindex 7959c598594eio_service52c953e3794f5379cf89

kind 5f986f4f4e674f504f4f:
c57972f456b6a3e5c4b66

- YCK_TCP_CLIENT
- YCK_TCP_SERVER
- YCK_UDP_CLIENT
- YCK_UDP_SERVER
- YCK_KCP_CLIENT
- YCK_KCP_SERVER

- YCK_SSL_CLIENT

io_service::close

51204F89628946854A8B89

```
void close(transport_handle_t transport);  
void close(int cindex);
```

5365
C278

transport

5C895185764F8F4A8B89

cindex

5C895185764F8F89

6C61
E88F

EB8E TCP 6F 5C4A5389409E74662C5898E6389

io_service::is_open

52A04F89624F8F4A8B8954694E525F726989

```
bool is_open(transport_handle_t transport) const;  
bool is_open(int cindex) const;
```

5365
C278

transport

4F8F4F8B536789
20531A00E3C482

cindex

4F89205F89

8F5650
B4DE3C

true: 5355FF false: 2A5355

io_service::dispatch

52	6D	7F	7E	7E	7A	4E	75	76	4E	4E	30
06	3E	51	DC	BF	0B	A7	1F	84	8B	F6	02

```
void dispatch(int max_count);
```

53	65
C2	70

max_count

67	6B	67	59	52	6D	4E	4E	65	30
2C	21	00	27	06	3E	8B	F6	70	02

6C	61
E8	0F

90 5E 6B 65 6C 5E 5F 57 51 53 7F 17 7E 4E 4E 76 4C 52 90 8F 7E 7A 8C 75 4F 59 Cocos2d-x 76 6E 67 7E 7A FF 4E 53 51 4E 6E 62 5F 64 (Unity, UE4) 76 4E 90 8F 7E 7A 30
1A 38 64 B9 D5 94 53 28 73 C3 51 DC 8B F6 84 1A A1 3B 91 BF 0B 03 28, 8B 82 84 32 D3 BF 0B 0C E5 CA 76 D6 38 0F 15 CE 84 3B 3B 91 BF 0B 02

6B	65	6C	5B	4E	5B	51	57	66	65	6E	62	75	97	97	5E	67	75	30
64	B9	D5	F9	8E	89	68	30	F4	B0	38	0F	4C	62	5E	38	09	28	02

794F
3A8B

```
yasio_shared_service()->dispatch(128);
```

io_service::write

54	4F	8F	4F	8B	8F	7A	53	90	65	63	30
11	20	93	1A	DD	DC	EF	D1	01	70	6E	02

```
int write(
    transport_handle_t thandle,
    std::vector<char> buffer,
    io_completion_cb_t completion_handler = nullptr
);
```

53	65
C2	70

thandle

4F	8F	4F	8B	53	67	30
20	93	1A	DD	E5	C4	02

buffer

89	53	90	76	4E	8F	52	7F	51	53	30
81	D1	01	84	8C	DB	36	13	B2	3A	02

```
completion_handler
```

53	90	5B	62	56	8C	30
D1	01	8C	10	DE	03	02

8F5650
04DE3C

645F53897662598276, < 0 : F86F637F198F89

6C61
E88F

completion_handler 4F2F63 KCP 39

7Abuffer4A796388F5756F4E464A8A63 completion_handler 89

io_service::write_to

54UDP28834A886389766289

```
int write_to(  
    transport_handle_t thandle,  
    std::vector<char> buffer,  
    const ip::endpoint& to,  
    io_completion_cb_t completion_handler = nullptr  
);
```

5365
2270

thandle

4F834A88E3C289

buffer

89538976buffer89

to

895389768F7A575789

completion_handler

618188166E8189

8F5650
04DE3C

765F538976598276, 55 < 0 F86F6F637F198F8F1A582F26834A88F273E5(TCP8E63A86678)89

6C61
E88F

63F07845E7Z84E DGRAM 28834A886F73 UDP, KCP 39

538988166E89 completion_handler 4F2F63 KCP 39

buffer completion_handler

io_service::schedule

```
highp_timer_ptr schedule(  
    const std::chrono::microseconds& duration,  
    timer_cb_t cb  
);
```

duration

cb

std::shared_ptr

```
// Register a once timer, timeout is 3 seconds.  
yasio_shared_service()->schedule(std::chrono::seconds(3), []()->bool{  
    printf("time called!\n");  
    return true;  
});  
  
// Register a loop timer, interval is 5 seconds.  
auto loopTimer = yasio_shared_service()->schedule(std::chrono::seconds(5), []()->bool{  
    printf("time called!\n");  
    return false;  
});
```

io_service::init_globals

```
static void init_globals(print_fn2_t print_fn);
```

```
print_fn
```

81	5B	4E	7F	7E	65	5F	62	53	51	65	30
FA	9A	49	51	DC	E5	D7	53	70	FD	70	02

6C	61
E8	0F

64 5D 68 6E 6F 63 69 83 78 6C 4F 6E 53 78 67 69 83 66 57 6C 63 57 61 5B 5A 51 53 2E 5A 4A 65 67 7B 7D 7E 63 6C
52 97 5E 5B 69 78 6C 8B 82 91 5A 54 38 UE4 54 U3D 84 65 5F 83 51 89

79	4F
3A	8B

```
// yasio_uelua.cpp
// compile with: /EHsc
#include "yasio_uelua.h"
#include "yasio/platform/yasio_ue4.hpp"
#include "lua.hpp"
#ifdef NS_SLUA
using namespace NS_SLUA;
#endif
#include "yasio/bindings/lyasio.cpp"

DECLARE_LOG_CATEGORY_EXTERN(yasio_ue4, Log, All);
DEFINE_LOG_CATEGORY(yasio_ue4);

void yasio_uelua_init(void* L)
{
    auto Ls = (lua_State*)L;
    print_fn2_t log_cb = [] (int level, const char* msg) {
        FString text(msg);
        const TCHAR* tstr = *text;
        UE_LOG(yasio_ue4, Log, L"%s", tstr);
    };
    io_service::init_globals(log_cb);

    luaregister_yasio(Ls);
}

void yasio_uelua_cleanup()
{
    io_service::cleanup_globals();
}
```

io_service::cleanup_globals

97 60 65 6C FF 66 79 57 6E 74 51 5C 65 63 30
59 01 B9 D5 0C 3E 3A 30 05 06 68 40 70 6E 02

```
static void cleanup_globals();
```


6C61
E80F

5378578081788F6354E15845E55F63786E8369A86958(.dll,.so)52E598837869F8656E8C9964787A5FE2A8869

io_service::channel_at

8A87EF83725F8762E189E36482

```
io_channel* channel_at(size_t cindex) const;
```

5365
C270

cindex

4F837265

8F5650
04DE3C

4F83E3646788, 53725F5880FA83F4F86E845E nullptr 82

io_service::set_option

887E889882

```
void set_option(int opt, ...);
```

5365
C270

opt

8898674E, 88E777 YOPT_X_XXX.

794F
3A8B

```
#include "yasio/yasio.hpp"

int main(){
    using namespace yasio;
    using namespace yasio::inet;
    io_hostent hosts[] = {
        {"192.168.1.66", 20336},
        {"192.168.1.88", 20337},
    };
    auto service = std::make_shared<io_service>(hosts, YASIO_ARRAYSIZE(hosts));
```

```
// for application protocol with length field, you just needs set this option.
// it's similar to java netty length frame based decode.
// such as when your protocol define as following
//     packet.header: (header.len=12bytes)
//         code:int16_t
//         datalen:int32_t (not contains packet.header.len)
//         timestamp:int32_t
//         crc16:int16_t
//     packet.data
service->set_option(YOPT_C_LFBFD_PARAMS,
    0,        // channelId, the channel index
    65535,    // maxFrameLength, max packet size
    2,        // lenghtFieldOffset, the offset of length field
    4,        // lengthFieldLength, the size of length field, can be 1,2,4
    12,       // lengthAdjustment. If the value of length feild ==
packet.header.len + packet.data.len, this parameter should be 0, otherwise should be
sizeof(packet.header)
);

// for application protocol without length field, just sets length field size to -1.
// then io_service will dispatch any packet received from server immediately,
// such as http request, this is default behavior of channel.
service->set_option(YOPT_C_LFBFD_PARAMS, 1, 65535, -1, 0, 0);
return 0;
}
```

8B5396
F7C205

[io_event Class](#)

[io_channel Class](#)

[io_service Options](#)

[xxsocket Class](#)

[obstream Class](#)

[ibstream_view Class](#)

[ibstream Class](#)

io_channel Class

TCP/SSL/UDP/KCP

```
namespace yasio { namespace inet { class io_channel; } }
```

Name	Description
io_channel::get_service	io_service
io_channel::index	
io_channel::remote_port	
io_channel::bytes_transferred	
io_channel::connect_id	ID

io_service
io_service::channel_at

io_channel::get_service

io_service

```
io_service& get_service()
```

io_channel::index

87624f998f7a7ae3

```
int index() const
```

io_channel::remote_port

87624f998f7a7ae3.

```
u_short remote_port() const;
```

8f5650
04de3c

8f564f998f7a7ae3f7

- f98e2377a4f998879981768f7a7ae3
- f98e6757a4f99887976547ae3

io_channel::bytes_transferred

87624f998f7a7ae3565982f8

```
long long bytes_transferred() const;
```

8f5650
04de3c

4e8e63fa7a5f58565520ff6336585982f86c784ef88a2377a636641c7

io_channel::connect_id

87624f998f7a7ae3565982f8id

```
unsigned int connect_id() const;
```

8f5650
04de3c

58677a4f9955208f63id



[io_service Class](#)

[io_event Class](#)

io_event Class

io_service

```
namespace yasio { namespace inet { class io_event; } }
```

Name	Description
io_event::kind	
io_event::status	
io_event::packet	
io_event::timestamp	
io_event::transport	
io_event::transport_id	ID
io_event::transport_udata	

io_event::kind

```
int kind() const;
```


io_event::transport

87524E4E794E854F8B23C789

```
transport_handle_t transport() const;
```

8F5650
D4DE3C

5F595367FF5F6252555F4E4E656F26834F8B5367593963FF4E53734F575759688F39
D4DE53C7FF5F6252555F4E4E656F26834F8B5367593963FF4E53734F575759688F39

io_event::transport_id

87524E4E794E854F8BID89

```
unsigned int transport_id() const;
```

8F5650
D4DE3C

324F6378F3727883F48579534EID89

io_event::transport_udata

887F1687524E854F8B7857656289

```
template<typename _Uty>  
_Uty io_event::transport_udata();  
  
template<typename _Uty>  
void io_event::transport_udata(_Uty uservalue);
```

6C61
E80F

7857978981F17874 userdata 785358, 8882:

- 65528E63FA7A62524E4E655858userdata
- 65528E6372594E4E656574userdata

8B5396
F7C205

[io_service Class](#)

ostream Class

634f4f8b525f175252f0b9

- 3.35.0 basic_ostream C++
- ostream int16~int64 float/double, ,
- fast_ostream

8b6c
ed05

```
namespace yasio {  
using ostream = basic_ostream<endian::network_convert_tag>;  
using fast_ostream = basic_ostream<endian::host_convert_tag>;  
}
```



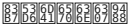


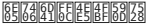
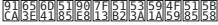
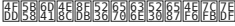
6254
1058

515167905165
6c718420fd70


Name	Description
ostream::ostream	ostream

5151656c
6c718905

Name	Description
ostream::write	
ostream::write_ix	(7bit Encoded Int/Int64)
ostream::write_v	(7bit Encoded Int)
ostream::write_byte	

Name	Description
obstream::write_bytes	
obstream::empty	
obstream::data	
obstream::length	
obstream::buffer	
obstream::clear	
obstream::shrink_to_fit	
obstream::save	



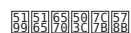
: obstream.hpp

obstream::obstream

 obstream 

```
obstream(size_t capacity = 128);  
  
obstream(const obstream& rhs);  
  
obstream(obstream&& rhs);
```

obstream::write



```
template<typename _Nty>  
void obstream::write(_Nty value);
```

5365
C270

value

8951517959
8159636430

6C61
E88F

_Nty 5849705757456948671~8598263637057626979705739

ostream::write_ix

6632/644F6968594E7Bit Encoded Int656F887F6451516989

```
template<typename _Intty>  
void ostream::write_ix(_Intty value);
```

5365
C270

value

895151795989
815963643082

6C61
E88F

_Intty 705753896959487057

- int32_t
- int64_t

64F0708879707069696F7C68AE8Fdotnet504E6165

- [BinaryWriter.Write7BitEncodedInt](#)
- [BinaryWriter.Write7BitEncodedInt64](#)

ostream::write_v

51514E8F5268626C6C63529F5E5788(7Bit Encoded Int).

```
void write_v(cxx17::string_view sv);
```

5365
C270

sv

89515176596238
81996364786282

6C61
E88F

69F17851497Bit Encoded7678636F51515162627F26, 608972 write_bytes 515159826362.

ostream::write_byte

515114A598238
89512A578282

```
void write_byte(uint8_t value);
```

5365
C270

value

895151765938
819963643C82

6C61
E88F

69F1785288784F4E ostream::write

ostream::write_bytes

51515982637C38
89635982787C82

```
void write_bytes(cxx17::string_view sv);  
  
void write_bytes(const void* data, int length);  
  
void write_bytes(std::streamoff offset, const void* data, int length);
```

5365
C270

sv

5151string_view5388685982637C.

data

8951515982637C7C7C985757.
8199635982787C7C7C985757.

length

8951515982657E7E79955E.
8169655782787C647F8A.

offset

8951515982657E7E79685979.
8169655782787C647E684F79.

6C61
E88F

offset + length 78505F985C4E
645C5F786F6E ostream::length

ostream::empty

5268496F524E7A82
24AD496F263A7A82

```
bool empty() const;
```

8F5650
04DE3C

true 7A; false 527A82

6C61
E88F

695168784F4E length == 082
697878494F6E

ostream::data

83536563639439
87D8786E878882

```
const char* data() const;
```

```
char* data();
```

8F5650
04DE3C

598260656388575739
598260656388575739

ostream::length

8353609F5E82
87D8497FA682

```
size_t length() const;
```

8F5650
04DE3C

8F56604F532476685982F539
64664126632864685782F539

ostream::buffer

8F56604F532476685982F539
64664126632864685782F539

```
const std::vector<char>& buffer() const;  
  
std::vector<char>& buffer();
```

8F5650
04DE3C

60642875632456236F22457F78
std::move 629069

794F
3A8B

```
// ostream_buffer.cpp  
// compile with: /EHsc  
#include "yasio/ostream.hpp"  
  
int main( )  
{  
    using namespace yasio;  
    using namespace cxx17;  
  
    ostream obs;  
    obs.write_v("hello world!");  
  
    const auto& const_buffer = obs.buffer();  
  
    // after this line, the obs will be empty  
    auto move_buffer = std::move(obs.buffer());  
  
    return 0;  
}
```

ostream::clear

6F7460FF4F4F587539
6566416C4F4F587539

```
void clear();
```

6C61
E80F

6BFD78651A2165buffer5158FF594E8A653759755F1753582758677583

ostream::shrink_to_fit

2A524065E815623A1A55655882

```
void shrink_to_fit();
```

ostream::save

5C6045764F8F52598278624F5852654F82

```
void save(const char* filename) const;
```

794F
3A8B

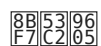
```
// ostream_save.cpp
// compile with: /EHsc
#include "yasio/ostream.hpp"
#include "yasio/istream.hpp"

int main( )
{
    using namespace yasio;
    using namespace cxx17;

    ostream obs;
    obs.write_v("hello world!");
    obs.save("ostream_save.bin");

    istream ibs;
    if(ibs.load("ostream_save.bin")) {
        // output should be: hello world!
        try {
            std::count << ibs.read_v() << "\n";
        }
        catch(const std::exception& ex) {
            std::count << "read_v fail: " <<
                << ex.message() << "\n";
        }
    }

    return 0;
}
```



[ibstream_view Class](#)

[ibstream Class](#)

ibstream_view Class

634f4f8b52725f57525f8032

6567

27c38f5716c77682b6c56959736246b3f34792fa std::out_of_range 6568

886c
ed05

```
namespace yasio {  
  // 206f575287768c4f81528f6259825f6c8925a8516c4685  
  using ibstream_view = basic_ibstream_view<endian::network_convert_tag>;  
  
  // 206f575287768c4f81528f6259825f6c8925a8516c4685  
  using fast_ibstream_view = basic_ibstream_view<endian::host_convert_tag>;  
}
```

6254
1058

62716728

Name	Description
ibstream_view::ibstream_view	672812a ibstream_view 6961

5151656c
6c7189b5

Name	Description
ibstream_view::reset	917f5f235ff21239362 2b6885c68f17167862
ibstream_view::read	7d7821776c88637838
ibstream_view::read_ix	7d7821776c8863 (7bit Encoded Int/Int64) 7d7838
ibstream_view::read_v	88625e772657 (7bit Encoded Int/Int64) 294e86327832

Name	Description
ibstream_view::read_byte	<code>1145982</code>
ibstream_view::read_bytes	<code>8853635892584881526563 F8D6879A7FA68C8B367862</code>
ibstream_view::empty	<code>2824415824517A</code>
ibstream_view::data	<code>83634178626788</code>
ibstream_view::length	<code>836341278F</code>
ibstream_view::advance	<code>545278534078F8638869</code>
ibstream_view::seek	<code>78A84184F8626569</code>

`6C61
E80F`

`ibstream_view 1924C++17892479 std::string_view, 8153775478585384235F71538F7A268D4FA7734E4E GC89`

`896C
8142`

`34874E: ibstream.hpp`

ibstream_view::ibstream_view

`6428462A ibstream_view 588C38`

```
ibstream_view();

ibstream_view(const void* data, size_t size);

ibstream_view(const obstream* obs);
```

`5365
C278`

data

`85285F575348885678628830575788`

size

5F C0 5F 17 53 4E 8F 52 63 62 59 6C 89

obs

F9 6F 17 53 76 61 89

istream_view::reset

21 7F istream_view 15 62 89 5F 89

```
void istream_view::reset(const void* data, size_t size);
```

53 65
C2 70

data

5F C0 5F 17 53 4E 8F 52 63 62 89 67 57 89

size

5F 53 5F 52 53 4E 8F 52 63 59 6C 89
85 C0 8F 17 16 8C 8B 36 76 6E 27 6F 82

istream_view::read

4E 60 4E 8B 53 63 58 89
CE 41 26 8B 06 76 5C 89

```
template<typename _Nty>  
_Nty istream_view::read();
```

8F 58 58
04 0E 3C

8F 58 8B 52 79 58
04 0E 8B 30 64 3C

6C 61
E8 0F

_Nty 58 89 79 8B 52 4E 68 4E 67 1~8 59 82 63 63 7C 52 62 6D 79 7C 52 89
8B 25 16 79 76 53 76 5C 89

istream_view::read_ix

8B 53 7Bit Encoded Int 53 25 16 79 76 53 76 5C 89

```
template<typename _Intty>  
_Intty istream_view::read_ix();
```


8F5650
04DE3C

32/644F63785089

6C61
E80F

_Intty 795F985F4E4E7C57

- int32_t
- int64_t

7C7D787C8B4E Microsoft dotnet 894E7D78

- [BinaryReader.Read7BitEncodedInt\(\)](#)
- [BinaryReader.Read7BitEncodedInt64\(\)](#)

ibstream_view::read_v

8B5353974E8F52636282

```
cxx17::string_view read_v();
```

8F5650
04DE3C

8F568B5352794E8F526363786E786E6E6E6E GC82

6C61
E80F

7C7D787A4B8B53 7bit Encoded Int827576797897A658FF6D8978 read_bytes 8B534E8F525982786282

ibstream_view::read_byte

8B5312A578282

```
uint8_t read_byte();
```

8F5650
04DE3C

uint8_t5082

6C61
E80F

675163784F4E ibstream_view::read

ibstream_view::read_bytes

FB62676A9F5A598263626C63 GC82

```
cxx17::string_view read_bytes();
```

8F5650
D4DE3C

4F6B52636274 cxx17::string_view 7B67636F62

ibstream_view::empty

52636069544E7A82
24AD412F263A7A82

```
bool empty() const;
```

8F5650
D4DE3C

true 7A; false 604E8159535414F598262
412BFB11632B12A598262

6C61
E80F

696363784F4E length == 082

ibstream_view::data

8F56606363639482
D46E41786E678882

```
const char* data() const;
```

8F5650
D4DE3C

8F6E6754694E78464A598276678882

istream_view::length

8752609275E39

```
size_t length() const;
```

8F5659
D4DE3C

5F52609275E39
53404177A682

istream_view::advance

545279526D8B536F6839
1140FBAB41FB66386782

```
void advance(ptrdiff_t offset);
```

5365
C278

offset

89545279527659799139
811140FBAB844FFBCE82

6C61
E86F

82*offset*4F8D65FF525364E379528B536F6839
261F78419CB1E3FBAB8B66386782

istream_view::seek

79528B536F6839
FBABFB66386782

```
ptrdiff_t seek(ptrdiff_t offset, int whence);
```

5365
C278

offset

84*whence*76517959799139
8173844FFBCE82

whence

544E78644EC68F63674E59FF SEEK_SET, SEEK_CUR, SEEK_END 82
2849496C8E

8F5650
04DE3C

8F5E78A85A78F84E6D885982597989

ibstream Class

634F4E8B525962528F54536E5753528882

8B6C
EDD5

```
namespace yasio {  
using ibstream = basic_ibstream<endian::network_convert_tag>;  
// F8F87F787A6CFF5F82FF8C3C6C7F8F8F  
using fast_ibstream = basic_ibstream<endian::host_convert_tag>;  
}
```

6254
1058

515167885165
6C718428FD78

Name	Description
ibstream::ibstream	849818E ibstream F88C

5151656C
6C7189D5

Name	Description
ibstream::load	FE68FE628F6D

7E625C6B7E67
E77F4221D384

ibstream_view

ibstream

ibstream::ibstream

6788464E ibstream F88C89

```
istream(std::vector<char> blob);  
  
istream(const ostream* obs);
```

`blob`

`obs`

`ostream`

`load`

`load(const char* filename) const;`

`load`

`load`

```
bool load(const char* filename) const;
```

`load`

`load`

`load`

`load`


`load`

`load`

`load`

xxsocket Class

bsd socket API yasio



```
xxsocket accept_n xxx_n socket
```

```
namespace yasio { namespace inet { class xxsocket; } }
```

Name	Description
xxsocket::xxsocket	xxsocket

Name	Description
xxsocket::xpconnect	TCP
xxsocket::xpconnect_n	TCP
xxsocket::pconnect	TCP
xxsocket::pconnect_n	TCP
xxsocket::pserve	tcp
xxsocket::swap	socket
xxsocket::open	socket

Name	Description
xxsocket::reopen	 socket
xxsocket::is_open	 socket
xxsocket::native_handle	 socket
xxsocket::release_handle	 socket
xxsocket::set_nonblocking	 socket
xxsocket::test_nonblocking	 socket
xxsocket::bind	 socket
xxsocket::bind_any	 socket
xxsocket::listen	 TCP
xxsocket::accept	 TCP
xxsocket::accept_n	 TCP
xxsocket::connect	 socket
xxsocket::connect_n	 socket
xxsocket::send	 socket
xxsocket::send_n	 socket
xxsocket::recv	 socket
xxsocket::recv_n	 socket
xxsocket::sendto	 DGRAM
xxsocket::recvfrom	 DGRAM

Name	Description
xxsocket::handle_write_ready	 xxsocket
xxsocket::handle_read_ready	 xxsocket
xxsocket::local_endpoint	 xxsocket
xxsocket::peer_endpoint	 xxsocket
xxsocket::set_keepalive	 tcp keepalive
xxsocket::reuse_address	 xxsocket
xxsocket::exclusive_address	 xxsocket
xxsocket::select	 xxsocket
xxsocket::shutdown	 xxsocket
xxsocket::close	 xxsocket
xxsocket::tcp_rtt	 tcp rtt.
xxsocket::get_last_errno	 xxsocket
xxsocket::set_last_errno	 xxsocket
xxsocket::strerror	 xxsocket
xxsocket::gai_strerror	 getaddrinfo
xxsocket::resolve	 xxsocket
xxsocket::resolve_v4	 ipv4
xxsocket::resolve_v6	 ipv6
xxsocket::resolve_v4to6	 ipv4V4MAPPED

Name	Description
<code>xxsocket::resolve_tov6</code>	<code>ipv4</code> <code>ipv6</code> <code>V4MAPPED</code>
<code>xxsocket::getipsv</code>	<code>ip</code>
<code>xxsocket::traverse_local_address</code>	

xxsocket::xxsocket

`xxsocket`

```
xxsocket::xxsocket();  
xxsocket::xxsocket(socket_native_type handle);  
xxsocket::xxsocket(xxsocket&& right);  
xxsocket::xxsocket(int af, int type, int protocol);
```

`handle`

`handle`

`socket` `xxsocket`

`right`

`move`

`af`

`ip`

`protocol`

`TCP/UDP` `0`

xxsocket::xpconnect

`TCP`

```
int xxsocket::xpconnect(const char* hostname, u_short port, u_short local_port = 0);
```

`hostname`

`hostname`

`IP` `16` `5760`

port

818f636752587a7af238

local_port

7c5a9a4f7af23f76c6b8a58 0 887a8f67529139

6c61
e80f

4f8152686b2c67626178ip538f68727c8

8f5658
b4be3c

0: 7252ff < 0 5129ff9a8f xxsocket::get_last_errno 8753938f7939

xxsocket::xpconnect_n

8f8c7a675258fa7aTCP8f6389

```
int xxsocket::xpconnect_n(const char* hostname, u_short port, const std::chrono::microseconds&
wtimeout, u_short local_port = 0);
```

5365
c270

hostname

818f636752586b6754ffef4e6f IP67a 16 5760 82

port

818f636752587a7af238

local_port

7c5a9a4f7af23f76c6b8a58 0 887a8f67529139

wtimeout

fa7a8f6389f8629539

8f5658
b4be3c

0: 7252ff < 0 5129ff9a8f xxsocket::get_last_errno 8753938f7939

6c61
e80f

4f8152686b2c67626178ip538f68727c8

xxsocket::pconnect

548C 8FD0 7A0B 670D 52A1 5668 5EFA 7ACB TCP 8FDE 63A5 3002

```
int xxsocket::pconnect(const char* hostname, u_short port, u_short local_port = 0);
int xxsocket::pconnect(const endpoint& ep, u_short local_port = 0);
```

53	65
C2	70

hostname

89 8F 63 67 52 56 4E 67 54 FF 53 4E 66
81 DE A5 0D A1 68 3B 3A 0D 0C EF E5 2F IP 57 57 62 57 54 30
81 DE A5 0D A1 68 3B 3A 0D 0C EF E5 2F 30 40 16 DF 0D 02

ep

89	8F	63	67	52	56	76	57	57	30
81	DF	A5	0D	A1	68	84	30	40	02

port

89	8F	63	67	52	56	76	7A	53	30
81	DF	A5	0D	A1	68	84	FF	E3	02

local_port

```
67 57 90 4F 7A 53 53 FF 9E 8B 50 0 88 79 96 67 52 91 30
2C 30 1A F1 FF F3 F7 AC D8 A4 3C 68 3A 8F 3A 06 4D 02
```

8F	56	50
D4	DE	3C

```
0: 62 52 FF < 0 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 30
   10 0F 0C 31 25 0C 1A C7 B7 D6 19 EF 01 02
```

6C61
F80F4E4F686D6767656376ip538B6830
0D1AC04B2C3A2F01844F4E0802

xxsocket::pconnect_n

```
548F7A6752565E7ATCP8F6330
8CDC0B0DA168FACBDFE502
```

```
int pconnect_n(const char* hostname, u_short port, const std::chrono::microseconds& wtimeout,
u_short local_port = 0);
int pconnect_n(const char* hostname, u_short port, u_short local_port = 0);
int pconnect_n(const endpoint& ep, const std::chrono::microseconds& wtimeout, u_short
local_port = 0);
int pconnect_n(const endpoint& ep, u_short local_port = 0);
```

53	65
C2	70

hostname

```
89 8F 63 67 52 56 4E 67 54 FF 53 4E 66
81 DF A5 0D A1 68 3B 3A 0D 0C FF F5 2F
```

ep

818f63675258795757b9

port

818f63675258797af3b9

local_port

6750904f7af3f76c6b8b50 0 88798f675291b9

wtimeout

fa7a8f6389f8f505b9

8f5650
b4de3c

0: 1052ff < 0 51296c1a87 xxsocket::get_last_errno 875393ef79b9

xxsocket::pserve

5f546757TCP6752795430
002f2c300da1d12c02

```
int pserve(const char* addr, u_short port);  
int pserve(const endpoint& ep);
```

5365
c270

addr

676a636a7f61 IP5a77 b9

ep

6767575739
2c5a5040b2

port

TCP79547af3b9

8f5650
b4de3c

0: 1052ff < 0 51296c1a87 xxsocket::get_last_errno 875393ef79b9

794f
3a8b

```
// xxsocket-serve.cpp  
#include <signal.h>  
#include <vector>  
#include "yasio/xxsocket.hpp"
```

```
using namespace yasio;
using namespace yasio::inet;

xxsocket g_server;
static bool g_stopped = false;
void process_exit(int sig)
{
    if (sig == SIGINT)
    {
        g_stopped = true;
        g_server.close();
    }
    printf("exit");
}
int main()
{
    signal(SIGINT, process_exit);

    if (g_server.pserve("0.0.0.0", 1219) != 0)
        return -1;
    const char reply_msg[] = "hi, I'm server\n";
    do
    {
        xxsocket cs = g_server.accept();
        if (cs.is_open())
        {
            cs.send(reply_msg, sizeof(reply_msg) - 1);
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
        }
    } while (!g_stopped);

    return 0;
}
```

xxsocket::swap

`xxsocket`

```
xxsocket& swap(xxsocket& who);
```

`who`

`xxsocket`

`xxsocket`

`xxsocket`

xxsocket::open

xxsocket

```
bool open(int af = AF_INET, int type = SOCK_STREAM, int protocol = 0);
```

af

AF_INET (ipv4) AF_INET6 (ipv6)

type

SOCK_STREAM (TCP), SOCK_DGRAM (UDP)

protocol

TCP/UDP 0

true: false xxsocket::get_last_errno

xxsocket::reopen

xxsocket

```
bool reopen(int af = AF_INET, int type = SOCK_STREAM, int protocol = 0);
```

af

AF_INET (ipv4) AF_INET6 (ipv6)

type

SOCK_STREAM (TCP), SOCK_DGRAM (UDP)

protocol

TCP/UDP 0

true: false xxsocket::get_last_errno

6C61
E80F

8267socket50635F6F6851654F5151F56F6B2168636F82

xxsocket::is_open

52A8socket2F24F2536682

```
bool is_open() const;
```

8F5650
D4DE3C

true: 2953666F false: 5A636682

xxsocket::native_handle

8368socket654F6F6F7882

```
socket_native_type native_handle() const;
```

8F5650
D4DE3C

socket6748636F786F yasio::inet::invalid_socket 887A2848socket82

xxsocket::release_handle

21636555socket6F6F78A7564362

```
socket_native_type release_handle() const;
```

8F5650
D4DE3C

21635274socket6748636F78

xxsocket::set_nonblocking

8876socket7482788585A6F82

```
int set_nonblocking(bool nonblocking) const;
```


5365
C270

nonblocking

true: 9798586A5F6F false: 9858215F62

8F5650
04DE3C

0: 6252FF < 0 5129FFA987 xxsocket::get_last_errno 875393287932

xxsocket::test_nonblocking

6840socket6F544E9798586A5F62

```
int test_nonblocking() const;
```

8F5650
04DE3C

1: 9798586A5F6F 0: 9858215F62

6C61
E80F

6840winsock26F544E9798586A5F62 SOCK_STREAM 7057socket1A645E -1 32

xxsocket::bind

7F5Asocket676767575739

```
int bind(const char* addr, unsigned short port) const;  
int bind(const endpoint& ep) const;
```

5365
C270

addr

676763587F53ip575739

port

817158737F5339

ep

897158737F53575739

8F5650
04DE3C

0: 1052FF < 0 5129FF9A07 xxsocket::get_last_errno 875393EF7039

xxsocket::bind_any

7F5Asocket276A4B61506089

```
int bind_any(bool ipv6) const;
```

5365
C270

ipv6

6F547F5B67674E61IPv65767
2F26616A2C3A4B0F

8F5650
04DE3C

0: 1052FF < 0 5129FF9A07 xxsocket::get_last_errno 875393EF7039

xxsocket::listen

6F2B705263EA TCP69577A63E746F96589

```
int listen(int backlog = SOMAXCONN) const;
```

5365
C270

backlog

6F5970546580

8F5650
04DE3C

0: 1052FF < 0 5129FF9A07 xxsocket::get_last_errno 875393EF7039

xxsocket::accept

6353464F5B627A8F6380

```
xxsocket accept() const;
```

8F5650
04DE3C

5458627A984F78 xxsocket FB8C39

xxsocket::accept_n

878858636F6A57664F58577A8E6389

```
int accept_n(socket_native_type& new_sock) const;
```

5365
C278

new_sock

83FAE2686C6458627A984F786554socketE3C455Z8

8F5650
04DE3C

0: 6252FF < 0 5928FF98CF xxsocket::get_last_errno 8753938F7889

6C61
E88F

82676857658F58 0 6Fnew_sock 4A28887E4E2798586A6F82

8978687A7848586CF8782A8385Z8 xxsocket::set_nonblocking 66socket887E4A2798586A6F82

xxsocket::connect

FA7A8E6389

```
int connect(const char* addr, u_short port);  
int connect(const endpoint& ep);
```

5365
C278

addr

8F7A4867ip575789

port

8F7A48677A5389

ep

8F7A4867575789

8F5650
04DE3C

0: f252ff < 0 5929ff1a2f xxsocket::get_last_errno 8753932ff7839

6061
e88f

TCP: 5189TCP 4f68e362

UDP: f57a43375758

xxsocket::connect_n

f57a8fa339
fa7a8fa339

```
int connect_n(const char* addr, u_short port, const std::chrono::microseconds& wtimeout);  
int connect_n(const endpoint& ep, const std::chrono::microseconds& wtimeout);  
int connect_n(const endpoint& ep);
```

5365
c298

addr

8f7a4667575739
8f7a4667575739

port

8f7a4667575739
8f7a4667575739

ep

8f7a4667575739
8f7a4667575739

wtimeout

f57a8fa33980f6f6f339
f57a8fa33980f6f6f339

8F5650
04DE3C

0: f252ff < 0 5929ff1a2f xxsocket::get_last_errno 8753932ff7839

6061
e88f

TCP: 5189TCP 4f68e362

UDP: f57a43375758

xxsocket::connect_n

socket 4

```
int disconnect() const;
```

```
0: < 0 xxsocket::get_last_errno
```

SOCK_DGRAM (UDP) socket

xxsocket::send

```
int send(const void* buf, int len, int flags = 0);
```

buf

len

flags

```
==len: < len xxsocket::get_last_errno
```

xxsocket::send_n

```
int send_n(const void* buf, int len, const std::chrono::microseconds& wtimeout, int flags = 0);
```

5365
C270

buf

89518863627680595982575739

len

895188636276955E39

wtimeout

618188F6F6F482

flags

618178625542678839

8F5650
04DE3C

```
==len: 6252FF < len 598DFF9A8F xxsocket::get_last_errno 835395887839
```

xxsocket::recv

AE516863298F7A4E6751888F67769656239

```
int recv(void* buf, int len, int flags = 0) const;
```

5365
C270

buf

A352786275B25A39

len

A352786275B25A725E39

flags

A35278625542678839

8F5650
04DE3C

```
==len: 6252FF < len 598DFF9A8F xxsocket::get_last_errno 835395887839
```

6C61
E88F

68F17869547A538F59FF53514Esocket678E695469 8288589A5E 39

xxsocket::recv_n

5480F2F2F2A1566A4E846862FA636A7F56766289

```
int recv_n(void* buf, int len, const std::chrono::microseconds& wtimeout, int flags = 0)
const;
```

5365
C270

buf

6369696275715389
A368706E75625A82

len

6369696275625A7F5689
A368706E75625A7F5689

wtimeout

63696962F2F2F289
A36880F2F2F289

flags

636969635E5C688889
A368706E9542678082

8F5050
04DE3C

```
==len: 6252FF < len 59896C7A8F xxsocket::get_last_errno 836399887889
```

xxsocket::sendto

54807A4E675389 DGRAM 68UDP69696289

```
int sendto(const void* buf, int len, const endpoint& to, int flags = 0) const;
```

5369
C270

buf

656189696275715389
856189706E75625A82

len

6561896962757153897F5689
856189706E75625A7F5689

to

61817068574089
81817068574089

flags

618170625542698889
818170625542698889

8F5650
04DE3C

```
==len: 6252FF < len 5989FF99FF xxsocket::get_last_errno 835395887839
```

xxsocket::recvfrom

4E815888248C7A4E675189276374986282

```
int recvfrom(void* buf, int len, endpoint& peer, int flags = 0) const;
```

5365
C278

buf

A358986275515A89

len

A358986275515A955682

peer

A35898626468FF85FA239889

flags

A35898625E5C688882

8F5650
04DE3C

```
==len: 6252FF < len 5989FF99FF xxsocket::get_last_errno 835395887839
```

6C61
E88F

68715859547A538458FF52514Esocket6788585468 828858255F 82

xxsocket::handle_write_ready

785Fsocket535182

```
int handle_write_ready(const std::chrono::microseconds& wtimeout) const;
```

5365
C278

wtimeout

785F88F8F8F882

8F5650
04DE3C

0: 80F3FF 1: F652FF < 0: 5189FF9A8F xxsocket::get_last_errno 876393887839

6C61
E80F

9A585F516861887F6251696F76C95146FF69F67674F7A53846889

xxsocket::handle_read_ready

7865socket278889

```
int handle_read_ready(const std::chrono::microseconds& wtimeout) const;
```

5365
C270

wtimeout

78658865F6F389

8F5650
04DE3C

0: 80F3FF 1: F652FF < 0: 5189FF9A8F xxsocket::get_last_errno 876393887839

xxsocket::local_endpoint

87634537499A4F766757505089

```
endpoint local_endpoint() const;
```

8F5650
04DE3C

8458675757575789

6C61
E80F

59676A6789788F xxsocket::connect 6288TCP8E633289E1626768626F9945846879575766 0.0.0.0

xxsocket::peer_endpoint

87634537499A4F76677A505089

```
endpoint peer_endpoint() const;
```

8F5658
04DE3C

8F566257575738
04DE2C30304082

6C61
E88F

5962666969728F xxsocket::connect 1689 TCP8E63369E31626268626E99458F587857572F 0.0.0.0

xxsocket::set_keepalive

887FTCP5E52518878C5F9C27682

```
int set_keepalive(int flag = 1, int idle = 7200, int interval = 75, int probes = 10);
```

5365
C278

flag

1 : 55546555524F88C5F9FF 0 : 71E582

idle

5351785C6C67464F6D60454E54FF54525E5C5388C5F8A2487867598D6868F46C534DFF79FF38
5351785C6C67464F6D60454E54FF54525E5C5388C5F8A2487867598D6868F46C534DFF79FF38

interval

5351626552C5F86654F86FCB586189C5F8A248A2F8F4F4846C534DFF68B26982

probes

5351626552C5F86654F86FCB586189C5F8A248A2F8F4F4846C534DFF68B26982

8F5658
04DE3C

0 : 188F6C < 0 5129FF9A8F xxsocket::get_last_errno 876893887882

794F
3A8B

```
// xxsocket-keepalive.cpp
#include "yasio/xxsocket.hpp"
using namespace yasio;
using namespace inet;

int main(){
    xxsocket client;
    if(0 == client.pconnect("192.168.1.19", 80)) {
```

```
        client.set_keepalive(1, 5, 10, 2);
    }
    return 0;
}
```

xxsocket::reuse_address

xxsocket::reuse_address

```
void reuse_address(bool reuse);
```

xxsocket::reuse_address

reuse

xxsocket::reuse_address

xxsocket::reuse_address

xxsocket::reuse_address

xxsocket::exclusive_address

xxsocket::exclusive_address

```
void exclusive_address(bool exclusive);
```

xxsocket::exclusive_address

exclusive

xxsocket::exclusive_address

xxsocket::exclusive_address

xxsocket::exclusive_address

xxsocket::set_optval

xxsocket::set_optval

```
template <typename _Ty> int set_optval(int level, int optname, const _Ty& optval);
```

5365
C270

level

socket899979A72B39

optname

8999797C5739
8979788882

optval

89995889

8F5658
04DE3C

0: 1252FF < 0 5129FF1A8F xxsocket::get_last_errno 8753958F7839

6C61
E80F

64F07854bsd socket setsockopt 528878546FE32F4F782A6750886CF4694F4F7539

xxsocket::get_optval

887Esocket899989

```
template <typename _Ty> _Ty get_optval(int level, int optname) const
```

5365
C270

level

socket899979A72B39

optname

8999797C5739
8979788882

8F5658
04DE3C

846E89995889

6C61
E80F

64F07854bsd socket getsockopt 528878546FE32F4F782A6750886CF4694F4F7539

xxsocket::select

7952socket846848484839

```
int select(fd_set* readfds, fd_set* writefds, fd_set* exceptfds, const
std::chrono::microseconds& wtimeout)
```

5365
C270

readfds

F2F84848F2F878657430

writefds

F2514848F2F878657430

exceptfds

55584848F2F878657430

wtimeout

7851484889F8484830

8F5650
D4DE3C

0:80F2FF > 0:6252FF < 0:5989FF908F xxsocket::get_last_errno 870819887930

xxsocket::shutdown

73E8TCP48851A8830

```
int shutdown(int how = SD_BOTH) const;
```

5365
C270

how

73E81A887988FFEF204E486A674E50

- SD_SEND : 51891A88
- SD_RECEIVE : 63631A88
- SD_BOTH : 88E873E8

8F5650
04DE3C

0: f252ff < 0: 5129ff992f xxsocket::get_last_errno 876293227932

xxsocket::close

7322socket6f2a62767f446662

```
void close(int shut_how = SD_BOTH);
```

5365
c270

shut_how

73221A9378576f6f2f454b4545674e52

- SD_SEND : 51899899
- SD_RECEIVE : 63539999
- SD_BOTH : 21282325
- SD_NONE : 58287325

8F5650
04DE3C

0: f252ff < 0: 5129ff992f xxsocket::get_last_errno 876293227932

6c61
e80f

5262 shut_how != SD_NONE 6f62f0781a488328 shutdown 6f518c786522 close 62

xxsocket::tcp_rtt

8752TCP8f6379RTT39

```
uint32_t tcp_rtt() const;
```

8F5650
04DE3C

8458TCP79RTTf2f2ff6c534b: 5e79 39

xxsocket::get_last_errno

xxsocket::get_last_errno

```
static int get_last_errno();
```

xxsocket::get_last_errno

```
0: < 0 ? 0 : xxsocket::strerror(errno)
```

xxsocket::get_last_errno

xxsocket::get_last_errno

xxsocket::set_last_errno

xxsocket::set_last_errno

```
static void set_last_errno(int error);
```

xxsocket::set_last_errno

error

xxsocket::set_last_errno

xxsocket::set_last_errno

xxsocket::set_last_errno

xxsocket::not_send_error

xxsocket::not_send_error

```
static bool not_send_error(int error);
```

xxsocket::not_send_error

error

xxsocket::not_send_error

8F5650
04DE3C

true : socket69586C false : socket7260F27E6179938BFC655373E6socket7E6299AF62

6C61
E80F

4E536389644F8F6E50 < 0 F6FC837869F07862

xxsocket::not_recv_error

7260F27E6179938BFC655373E6socket7E6299AF62

```
static bool not_recv_error(int error);
```

5365
C270

error

79EF7862

8F5650
04DE3C

true : socket69586C false : socket7260F27E6179938BFC655373E6socket7E6299AF62

6C61
E80F

4E536363644F8F6E50 < 0 F6FC837869F07862

xxsocket::strerror

5C958B788F634E597B4E30
8619EE616C623A57263262

```
static const char* strerror(int error);
```

5365
C270

error

79EF7862

8F5650
04DE3C

79EF7F697059784E30
8619EE616C623A57263262

xxsocket::gai_strerror

`getaddrinfo`

```
static const char* gai_strerror(int error);
```

`error`

error

`error`

`error`

`error`

xxsocket::resolve

`resolve`

```
int resolve(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port = 0,
            int socktype = SOCK_STREAM);
```

`endpoints`

endpoints

`endpoints`

hostname

`hostname`

port

`port`

socktype

`socktype`

`socktype`

`0: 0 > 0 0 xxsocket::strerror 0`

xxsocket::resolve_v4

IPv4

```
int resolve_v4(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port = 0, int socktype = SOCK_STREAM);
```

5365
C270

endpoints

8F51536539
83FA C27039

hostname

575439
6F6D62

port

7A5339
7F5339

socktype

socket795739
796D62

8F5659
646E3C

0: 68938F > 0 9A8F xxsocket::strerror 8F634A887F95884F6939

xxsocket::resolve_v6

IPv6

```
int resolve_v6(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port = 0, int socktype = SOCK_STREAM);
```

5365
C270

endpoints

8F51536539
83FA C27039

hostname

575439
6F6D62

port

7A5339
7F5339

socktype

socket7957397b8b62

8f5650d4de3c

0: 636366ff > 0 9a2f xxsocket::strerror 8f634a8876f3884f6939

xxsocket::resolve_v4to6

4f89675754535476c5e3606f67652b24IPv457575f8f634f3040766c623aIPv6 V4MAPPED685f393c6f62

```
int resolve_v4to6(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port = 0, int socktype = SOCK_STREAM);
```

5365c270

endpoints

8f5153653983faC27062

hostname

5754396f6062

port

7af339efef39

socktype

socket7957397b8b62

8f5650d4de3c

0: 636366ff > 0 9a2f xxsocket::strerror 8f634a8876f3884f6939

xxsocket::resolve_tov6

89675754535476c5e3606f67652b24c5e3606f67652b24IPv457574f8f634f3040766c623aIPv6 V4MAPPED685f393c6f62

```
int resolve_tov6(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port = 0, int socktype = SOCK_STREAM);
```

53	65
C2	70

endpoints

8F	51	53	65	30
93	FA	C2	70	02

hostname

57	54	30
DF	0D	02

port

7A	53	30
EF	E3	02

socktype

socket^{7C}_{7B}⁵⁷_{8B}³⁰₀₂

8F	56	50
D4	DE	3C

```
0: 65 95 88 FF > 0 90 8F xxsocket::strerror 8F 63 4E 8B 7E 95 19 8B 4F 60 30
E0 19 EF 0C 1A C7 6C 62 3A E6 C6 19 EF E1 6F 02
```

xxsocket::getipsv

83 53 67 67 65 63 76 IP 53 88 68 68 5F 4F 30
B7 D6 2C 3A 2F 01 84 4F AE 08 07 D7 4D 02

```
static int getipsv();
```

8F	56	50
D4	DE	3C

- `ipsv_ipv4`: `676767676767`|`Pv4``548889`
- `ipsv_ipv6`: `676767676767`|`Pv6``548889`
- `ipsv_dual_stack`: `676767676767`|`Pv4``54`|`Pv6``5468488889`

6C	61
E8	0F

```
5F 8F 56 50 65 63 53 68 53 8B 66 FF 75 67 5E 5F 59 7E 4F 51 4F 75 IPv4 90 4F FF
53 04 DE 3C 2F 01 CC 08 4F AE 2F 0C 28 37 94 53 CB C8 18 48 7F 28 1A E1 0C
```

4F 59 66 80 62 67 88 59 57 54 65 5F 54 8C 87 7A 7F 7E 65 FF 5C 4F 4F 51 90 62 88 82 7A FD 4B 3A BE 07 28 0C F6 00 2F wifi 8C 02 9D 51 DC F6 0C 06 1A 18 48 09 E9 wifi 0C

📶wifi📶IPv4: <https://github.com/halx99/yasio/issues/130>

79	4F
3A	8B

```
// xxsocket-ipsv.cpp
#include <vector>
#include "yasio/xxsocket.hpp"
using namespace yasio;
using namespace yasio::inet;
```

```
int main(){
    const char* host = "github.com";
    std::vector<ip::endpoint> eps;
    int flags = xxsocket::get_ipsv();
    if(flags & ipsv_ipv4) {
        xxsocket::resolve_v4(eps, host, 80);
    }
    else if(flags & ipsv_ipv6) {
        xxsocket::resolve_tov6(eps, host, 80);
    }
    else {
        std::cerr << "Local network not available!\n";
    }
    return 0;
}
```

xxsocket::traverse_local_address

674E6767575769
9A3E676A564069

```
static void traverse_local_address(std::function<bool(const ip::endpoint&)> handler);
```

5365
C278

handler

674E5757568C38
9A3E30400E6362

794F
3A8B

```
// xxsocket-traverse.cpp
#include <vector>
#include "yasio/xxsocket.hpp"
using namespace yasio;
using namespace yasio::inet;
int main(){
    int flags = 0;
    xxsocket::traverse_local_address([&](const ip::endpoint& ep) -> bool {
        switch (ep.af())
        {
            case AF_INET:
                flags |= ipsv_ipv4;
                break;
            case AF_INET6:
                flags |= ipsv_ipv6;
                break;
        }
        return (flags == ipsv_dual_stack);
    });
    YASIO_LOG("Supported ip stack flags=%d", flags);
}
```

```
    return flags;  
}
```

8B	53	96
F7	C2	05

[io_service Class](#)

io_service options

io_service

Name	Description
<i>YOPT_S_DEFER_EVENT_CB</i>	Set defer event callback params: callback:defer_event_cb_t remarks: a. User can do custom packet resolve at network thread, such as decompress and crc check. b. Return true, io_service will continue enqueue to event queue. c. Return false, io_service will drop the event.
<i>YOPT_S_DEFERRED_EVENT</i>	Set whether deferred dispatch event, default is: 1 params: deferred_event:int(1)
<i>YOPT_S_RESOLV_FN</i>	Set custom resolve function, native C++ ONLY params: func:resolv_fn_t*
<i>YOPT_S_PRINT_FN</i>	Set custom print function native C++ ONLY parmas: func:print_fn_t remarks: you must ensure thread safe of it
<i>YOPT_S_PRINT_FN2</i>	Set custom print function with log level parmas: func:print_fn2_t you must ensure thread safe of it
<i>YOPT_S_EVENT_CB</i>	Set event callback params: func:event_cb_t*
<i>YOPT_S_TCP_KEEPALIVE</i>	Set tcp keepalive in seconds, probes is tries. params: idle:int(7200), interal:int(75), probes:int(10)
<i>YOPT_S_NO_NEW_THREAD</i>	Don't start a new thread to run event loop. params: value:int(0)
<i>YOPT_S_SSL_CACERT</i>	Sets ssl verification cert, if empty, don't verify. params: path:const char*

Name	Description
<i>YOPT_S_CONNECT_TIMEOUT</i>	Set connect timeout in seconds. params: connect_timeout:int(10)
<i>YOPT_S_DNS_CACHE_TIMEOUT</i>	Set dns cache timeout in seconds. params: dns_cache_timeout : int(600),
<i>YOPT_S_DNS_QUERIES_TIMEOUT</i>	Set dns queries timeout in seconds, default is: 5000. params: dns_queries_timeout : int(5000) remark: a. this option must be set before 'io_service::start' b. only works when have c-ares c. since v3.33.0 it's milliseconds, previous is seconds. d. the timeout algorithm of c-ares is complicated, usually, by default, dns queries will failed with timeout after more than 75 seconds. e. for more detail, please see: https://c-ares.haxx.se/ares_init_options.html
<i>YOPT_S_DNS_QUERIES_TRIES</i>	Set dns queries tries when timeout reached, default is: 5. params: dns_queries_tries : int(5) remarks: a. this option must be set before 'io_service::start' b. relative option: <i>YOPT_S_DNS_QUERIES_TIMEOUT</i>
<i>YOPT_S_DNS_DIRTY</i>	Set dns server dirty. params: reserved : int(1) remarks: a. this option only works with c-ares enabled b. you should set this option after your mobile network changed
<i>YOPT_C_LFBFD_FN</i>	Sets channel length field based frame decode function. params: index:int, func:decode_len_fn_t* remark: native C++ ONLY
<i>YOPT_C_LFBFD_PARAMS</i>	Sets channel length field based frame decode params. params: index:int, max_frame_length:int(10MBytes), length_field_offset:int(-1), length_field_length:int(4), length_adjustment:int(0),

Name	Description
<i>YOPT_C_LFBFD_IBTS</i>	Sets channel length field based frame decode initial bytes to strip. params: index:int, initial_bytes_to_strip:int(0)
<i>YOPT_C_REMOTE_HOST</i>	Sets channel remote host. params: index:int, ip:const char*
<i>YOPT_C_REMOTE_PORT</i>	Sets channel remote port. params: index:int, port:int
<i>YOPT_C_REMOTE_ENDPOINT</i>	Sets channel remote endpoint. params: index:int, ip:const char*, port:int
<i>YOPT_C_LOCAL_HOST</i>	Sets local host for client channel only. params: index:int, ip:const char*
<i>YOPT_C_LOCAL_PORT</i>	Sets local port for client channel only. params: index:int, port:int
<i>YOPT_C_LOCAL_ENDPOINT</i>	Sets local endpoint for client channel only. params: index:int, ip:const char*, port:int
<i>YOPT_C_MOD_FLAGS</i>	Mods channel flags. params: index:int, flagsToAdd:int, flagsToRemove:int
<i>YOPT_C_ENABLE_MCAST</i>	Enable channel multicast mode. params: index:int, multi_addr:const char*, loopback:int
<i>YOPT_C_DISABLE_MCAST</i>	Disable channel multicast mode. params: index:int
<i>YOPT_C_KCP_CONV</i>	The kcp conv id, must equal in two endpoint from the same connection. params: index:int, conv:int
<i>YOPT_T_CONNECT</i>	Change 4-tuple association for io_transport_udp. params: transport:transport_handle_t remark: only works for udp client transport
<i>YOPT_T_DISCONNECT</i>	Dissolve 4-tuple association for io_transport_udp. params: transport:transport_handle_t remark: only works for udp client transport

Name	Description
<code>YOPT_B_SOCKOPT</code>	Sets io_base sockopt. params: io_base*,level:int,optname:int,optval:int,optlen:int

8B5396
F7C205

[io_service Class](#)

API

54547A95
7D0D7AF4

```
namespace yasio {}
```

5151656C
6C71B9D5

Name	Description
yasio::host_to_network	4B6A59825F8C717E59825F
yasio::network_to_host	717E59825F8C717E59825F
yasio::xhighp_clock	876A6362A7F6F433
yasio::highp_clock	876A6362A7F6F433
yasio::clock	876A6362A7F6F433
yasio::set_thread_name	8B7E8C23887F7A5A
yasio::basic_strerror	686F72592B45

yasio::host_to_network

4B6A59825F8C717E59825F82

59654E
3487F6

yasio/detail/endian_portable.hpp

```
template <typename _Ty>  
inline _Ty host_to_network(_Ty value);  
inline int host_to_network(int value, int size);
```

5365
c270

value

818f637d7950ffvalue7657_Ty 2345294f011~859827d507c5739

size

818f637d7950ff3c694857827d6cfafd2f1~4598239

8f5650
64be3c

7f7e59825f7d5039

794f
3a8b

```
#include <stdio.h>
#include <inttypes.h>
#include "yasio/detail/endian_portable.hpp"
int main(){
    uint16_t v1 = 0x1122;
    uint32_t v2 = 0x11223344;
    uint64_t v3 = 0x1122334455667788;
    // output will be: net.v1=2211, net.v2=44332211, net.v3=8877665544332211
    printf("net.v1=%04" PRIx16 " ", net.v2=%08" PRIx32 " ", net.v3=%016" PRIx64 "\n",
        yasio::host_to_network(v1),
        yasio::host_to_network(v2),
        yasio::host_to_network(v3));
    return 0;
}
```

yasio::network_to_host

7f7e59825f7d5039

59654f
3487f6

yasio/detail/endian_portable.hpp

```
template <typename _Ty>
inline _Ty network_to_host(_Ty value);
inline int network_to_host(int value, int size);
```

5365
c270

value

818f637d7950ffvalue7657_Ty 2345294f011~859827d507c5739

size

```
89 8F 63 76 65 50 67 65 58 82 65 FF 53 80 66 1~4 58 82 30
81 6C 62 84 70 3C 09 48 57 82 70 0C EA FD 2F 57 82 02
```

8F	56	50
D4	DE	3C

4E	67	5B	82	5E	65	50	30
3B	3A	57	82	8F	70	3C	02

79	4F
3A	8B

```
#include <stdio.h>
#include <inttypes.h>
#include "yasio/detail/endian_portable.hpp"
int main(){
    uint16_t v1 = 0x2211;
    uint32_t v2 = 0x44332211;
    uint64_t v3 = 0x8877665533442211;
    // output will be: net.v1=1122, net.v2=11223344, net.v3=1122334455667788
    printf("host.v1=%04" PRIx16 " ", host.v2=%08" PRIx32 " ", host.v3=%016" PRIx64 " \n",
        yasio::network_to_host(v1),
        yasio::network_to_host(v2),
        yasio::network_to_host(v3));
    return 0;
}
```

```
yasio::xhighp_clock
```

83	53	7E	79	7E	65	95	62	30
B7	D6	B3	D2	A7	F6	F4	33	02

59	65	4E
34	87	F6

```
yasio/detail/utis.hpp
```

```
template <typename _Ty = steady_clock_t>
inline highp_time_t xhighp_clock();
```

6A	67	53	65
21	7F	C2	70

 $_{-Ty}$

- `yasio::steady_clock_t`: 84 6E 55 5D 78 7F 68 24 63 73 79 66 24 63
- `yasio::system_clock_t`: 84 6E 7C 7F UTC 66 24 63

8F5650
040E3C

7E797F63F36389

yasio::highp_clock

8762AE7379F2F36389

59654E
3487F6

yasio/detail/utils.hpp

```
template <typename _Ty = steady_clock_t>  
inline highp_time_t highp_clock();
```

6A675365
217FC270

_Ty

- yasio::steady_clock_t: 8F5650527C7E63F3638979F2F363
- yasio::system_clock_t: 846E7B7EUTC F2F363

8F5650
040E3C

7E797F63F36389

yasio::clock

8762AE7379F2F36389

59654E
3487F6

yasio/detail/utils.hpp

```
template <typename _Ty = steady_clock_t>  
inline highp_time_t clock();
```

6A675365

_Ty

- yasio::steady_clock_t: 846E53527C7E63F3638979F2F363

- `yasio::system_clock_t`: 8A5B7C7EUTC F2F453

8F5659
04DE3C

EB797E65256238
62A7F6F43382

yasio::set_thread_name

8B7B8573897E7A6439
6E62A7F6F43382

59654E
3487F6

yasio/detail/thread_name.hpp

```
inline void set_thread_name(const char* name);
```

5365
C278

name

87898B7E7E7A5479
06818E6E6F6B6DFA

6C61
E80F

5165754E8B65597E7A7A5E975E677538
64FD78217F8ECAAD1A6F6B6B8F5E38692862

yasio::basic_strfmt

51656A67FF685F535B7B4E6289585B7B4E38
FD78217F8E6C6C6F16572B3216898D5B7B3262

59654E
3487F6

yasio/detail/strfmt.hpp

```
template <class _Elem, class _Traits = std::char_traits<_Elem>,  
         class _Alloc = std::allocator<_Elem>>  
inline std::basic_string<_Elem, _Traits, _Alloc> basic_strfmt(size_t n, const _Elem*  
format, ...);
```

5365
2270

n

5758buffer5959
1058

format

686F587843FF54687C655 printf 7864

8F5650
04DE3C

8F68685F536278 std::string 7C5759784E39
046E6C6F106E64

6C61
E80F

68994F73FFEF78634F78597E584859785254
0E957F786CE7F8634F78597E5848597842668

- yasio::strfmt: 686F53597865
- yasio::wcsfmt: 686F5358597865

794F
3A8B

```
#include "yasio/detail/strfmt.hpp"
int main() {
    std::string str1 = yasio::strfmt(64, "My age is %d", 19);
    std::wstring str2 = yasio::wcsfmt(64, L"My age is %d", 19);
    return 0;
}
```



yasio yasio yasio/detail/config.hpp

Name	Description
YASIO_HAVE_KCP	kcpkcp
YASIO_HEADER_ONLY	yasio
YASIO_SSL_BACKEND	SSLSSL OpenSSL/MbedTLS 3.36.0 (YASIO_HAVE_SSL) 1 (OpenSSL) 2 (mbedtls)
YASIO_ENABLE_UDS	unix domain socket unix win10 RS5+
YASIO_HAVE_CARES	c-ares c-ares yasioDNS 10 c-ares
YASIO_VERBOSE_LOG	
YASIO_NT_COMPAT_GAI	Windows XP getaddrinfo API
YASIO_USE_SPSC_QUEUE	SPSC io_service::write
YASIO_USE_SHARED_PACKET	std::shared_ptr
YASIO_HAVE_HALF_FLOAT	half.hpp
YASIO_DISABLE_OBJECT_POOL	
YASIO_DISABLE_CONCURRENT_SINGLETON	

yasio 7C535974
98050406

yasio 76706853597446468458TCP 6F694EUDP 6C826763897A697E63618067526F4E6945786476656F597439734F6582674E79695F:

- 1A2Fio_service0909 YOPT_C_LFBFD_PARAMS 8B7F4F99536539
- 1A2Fio_service0909 YOPT_C_LFBFD_FN 8B7F815845639F5E8361F078 decode_len_fn_t 09

6C61E80F

8158456897637F46705878F8F554E598269468863int58F86C455846897F78678845A806C647363F8F8863ARM87775982F980E288 SIGBUS 6258E28882F245C289817FE961637F467078828 io_channel::__builtin_decode_len 09

YOPT_C_LFBFD_PARAMS

8B7F4F997462F36289

5365C278

max_frame_length

67595397F5E6F880756893A5E586389

length_field_offset

637F46696978F84E68686368628859824FF889

length_field_length

637F46696959686C62631~489829298(uint8_t,uint16_t,uint24_t,int32_t)89

length_adjustment

637F466C645C88995859784F787E58825E7F7838886777517F89785395585873 0 6C520124A 8869597F5E 09

6C61E80F

886865397F465768C578787E28510C57825E7F7838886777517F89785395585873:

```
int io_channel::__builtin_decode_len(void* d, int n)
{
    int loffset = uparams_.length_field_offset;
    int lsize   = uparams_.length_field_length;
    if (loffset >= 0)
    {
        int len = 0;
        if (n >= (loffset + lsize))
```

```
{
    ::memcpy(&len, (uint8_t*)d + loffset, lsize);
    len = yasio::network_to_host(len, lsize);
    len += uparams_.length_adjustment;
    if (len > uparams_.max_frame_length)
        len = -1;
}
return len;
}
return n;
}
```

decode_len_fn_t

41 93 93 73 68 68 53 9f 56 fd 78 6f 57 88 89

```
typedef std::function<int(void* d, int n)> decode_len_fn_t;
```

53 65
c2 70

d

65 93 93 63 63 63 88 58 82 57 57 89

n

6f 93 93 63 63 9f 56 89

8f 56 50
84 6e 3c

84 6e 88 6f 53 78 63 6e 99 9f 56 89

- > 0 : 93 78 63 9f 56 63 57 89
- == 0 : 43 62 78 63 46 89 4f 93 78 88 6f 53 68 99 9f 56 6f yasio 55 42 4a 7f 7e 63 63 63 6f 48 63 63 52 63 63 63 6f 4f 6d 21 83 73 62 51 78 89
- < 0 : 93 78 63 9f 56 62 56 6c 4f 29 63 55 52 48 85 4f 88 63 66 89

6c 61
e8 0f

93 78 88 6f 53 9f 56 fd 78 6f 57 88 89 7e 7a 58 51 76 fd 4f 59 f9 4e Lua 19 46 2f 63 88 7e 2a 5a 49 93 78 88 6f 53 9f 56 fd 78 89

yasio Interop

Unity C# yasio C: https://github.com/yasio/yasio/blob/master/yasio/bindings/yasio_ni.cpp

OpenNSM2

Unity yasio-3.37.1 NetworkServiceManager unity: <https://github.com/yasio/OpenNSM2>

FAQ

91709598897B
CD89EE98E354

yasio 2f2446987a168757685?

yasio 7d98c5fe5799f884ef3f804ef794f552c695955fc4ef4f804ef4f8f51904f884f5f595c7478777f514f4f8f2189597579125717f55ffxxsocketfc969yasio7d98c5fe5799f884ef3f804ef794f552c695955fc4ef4f804ef4f8f51904f884f5f595c7478777f514f4f8f2189597579125717f55ffboost.asio82

yasio? 4545454f755Ac0487728

- asio, libevent, libev, libuv 6c464ef634ef4f8f2f92f5f7487473985f12fio697329576c74f6f6475f395524f723iocp, kqueue, epoll, select 78f9876f7f735295974747f6f68f6374764fTCP7d98c5fe5799f884ef3f804ef794f552c695955fc4ef4f804ef4f8f51904f884f5f595c7478777f514f4f8f2189597579125717f55ff
- yasio 6c8f637a74f6fTCPc263f95f98524f555269
- yasio 5cTCP, UDP, KCP7ef4668c1f2Transportf952654ef7f2869
- yasio f9f82f75f6c26275f3274f72select7a16875

yasio 5f262f6397995857548967?

5f61fe4f87894f68c-ares5ef6f854f552f885f5e73DNS5f78f6f7f5f74f5f87d45d7387select5a5a765647f73f6f676967f8865f76a895967465652fc-ares6fyasio51284f4fc999676d89676768684ef4f8f887555106282f6f9245f5f67a2cf568768298974f89

yasio 2f2446987a168757685 SSL/TLS 4685?

5f61fe4f87894f68cOpenSSL7d89MbedTLS89

io_event 594f4ef6255ce 26887f548353 userdata

8853831fhttps://github.com/yasio/ftp_server/blob/master/ftp_server.cpp#L98

yasio 69545974 SIGPIPE 4f53ff 2f262468 f1f71f

4e 3.30 78276f5f9f697439

826246597a6f6f6f82fios527a79f8966688f9f2a2263SIGPIPE7d98c5fe5799f884ef3f804ef794f552c695955fc4ef4f804ef4f8f51904f884f5f595c7478777f514f4f8f2189597579125717f55ffTCP8f63266f6f6f32f6f32f6f4f69: Broken pipe
88c1: https://github.com/yasio/yasio/issues/170

ios **ec=65, detail:No route to host**

- **ios**:
 - **APP** **ios14.1+** **Local Network Permission**
 - **10161** **SNMP via TLS**
- **APP** **49152~65535**
- **https://www.speedguide.net/port.php**

macOS **UDP** **40** **Message too long**

- **macOS** **UDP** **9126**
- **socket** **UDP**:
 - **xxsocket**: **sock_udp.set_optval(SOL_SOCKET, SO_SNDBUF, (int)65535);**
 - **io_service**: **https://github.com/yasio/yasio/blob/master/tests/speed/main.cpp#L223**

io_service **schedule**

io_service

std::thread

https://github.com/yasio/yasio/issues/244

xxsocket **_nodelay**

_nonblock **internal**

YOPT_TCP_KEEPALIVE

https://github.com/yasio/yasio/issues/117

Lua

• c++11:

- `kaguya` `c++` `lua_newuserdata`, `placement new`
`xcode clang release`
- `bing.com` `LUA_USER_ALIGNMENT_T=max_align_t`
- `max_align_t` `xcode clang` `long double` `16`
- <http://lua-users.org/lists/lua-l/2019-07/msg00197.html>

• c++14:

- `sol2` `sol2` `kaguya` `sol2` `lua_newuserdata`
`clang release`

• c++17:

- `sol2` `xcode` `ios11` `C++17` `new` `Aligned allocation/deallocation`
`-faligned-allocation`
- `ios10` `std::shared_mutex` `yasio` `Apple` `pthread`
- `Lua` `double` `max_align_t` `C11` https://en.cppreference.com/w/c/types/max_align_t `Lua` `ANSI C89` `LUA_USER_ALIGNMENT_T`
- `C++11` `std::max_align_t` https://en.cppreference.com/w/cpp/types/max_align_t

`max_align_t` `__stddef_max_align_t.h`

Can't load xxx.bundle on macOS?

The file `xxx.bundle` needs change attr by command `sudo xattr -r -d com.apple.quarantine xxx.bundle`

xxsocket resolve socketype

- `winsock`
- `0` `IP`

`F41A38C1EE98`

`F7C265 987E5E52F3`