

Contents

Foreword	vi
1 Scope	1
2 Normative references	2
3 Terms and definitions	3
4 General Principles	4
4.1 Conformance	4
4.2 Acknowledgments	4
5 Namespaces and headers	5
6 Future plans (Informative)	6
7 Feature test macros (Informative)	7
8 Method of description (Informative)	8
8.1 Structure of each clause	8
8.2 Other conventions	8
9 Error reporting	9
9.1 Synchronous operations	9
9.2 Asynchronous operations	10
9.3 Error conditions	10
9.4 Suppression of signals	10
10 Library summary	11
11 Convenience header	13
11.1 Header <experimental/net> synopsis	13
12 Forward declarations	14
12.1 Header <experimental/netfwd> synopsis	14
13 Asynchronous model	16
13.1 Header <experimental/executor> synopsis	16
13.2 Requirements	19
13.3 Class template <code>async_result</code>	27
13.4 Class template <code>async_completion</code>	28
13.5 Class template <code>associated_allocator</code>	29
13.6 Function <code>get_associated_allocator</code>	30
13.7 Class <code>execution_context</code>	30
13.8 Class <code>execution_context::service</code>	32
13.9 Class template <code>is_executor</code>	33
13.10 Executor argument tag	33
13.11 <code>uses_executor</code>	34

13.12	Class template <code>associated_executor</code>	34
13.13	Function <code>get_associated_executor</code>	35
13.14	Class template <code>executor_binder</code>	36
13.15	Function <code>bind_executor</code>	39
13.16	Class template <code>executor_work_guard</code>	40
13.17	Function <code>make_work_guard</code>	41
13.18	Class <code>system_executor</code>	42
13.19	Class <code>system_context</code>	43
13.20	Class <code>bad_executor</code>	44
13.21	Class <code>executor</code>	45
13.22	Function <code>dispatch</code>	49
13.23	Function <code>post</code>	50
13.24	Function <code>defer</code>	51
13.25	Class template <code>strand</code>	52
13.26	Class template <code>use_future_t</code>	56
13.27	Partial specialization of <code>async_result</code> for <code>packaged_task</code>	59
14	Basic I/O services	61
14.1	Header <code><experimental/io_context></code> synopsis	61
14.2	Class <code>io_context</code>	61
14.3	Class <code>io_context::executor_type</code>	65
15	Timers	67
15.1	Header <code><experimental/timer></code> synopsis	67
15.2	Requirements	67
15.3	Class template <code>wait_traits</code>	68
15.4	Class template <code>basic_waitable_timer</code>	69
16	Buffers	73
16.1	Header <code><experimental/buffer></code> synopsis	73
16.2	Requirements	78
16.3	Error codes	82
16.4	Class <code>mutable_buffer</code>	82
16.5	Class <code>const_buffer</code>	83
16.6	Buffer type traits	84
16.7	Buffer sequence access	85
16.8	Function <code>buffer_size</code>	85
16.9	Function <code>buffer_copy</code>	85
16.10	Buffer arithmetic	86
16.11	Buffer creation functions	86
16.12	Class template <code>dynamic_vector_buffer</code>	88
16.13	Class template <code>dynamic_string_buffer</code>	89
16.14	Dynamic buffer creation functions	91
17	Buffer-oriented streams	92
17.1	Requirements	92
17.2	Class <code>transfer_all</code>	94
17.3	Class <code>transfer_at_least</code>	95
17.4	Class <code>transfer_exactly</code>	95
17.5	Synchronous read operations	96
17.6	Asynchronous read operations	98

17.7	Synchronous write operations	99
17.8	Asynchronous write operations	100
17.9	Synchronous delimited read operations	102
17.10	Asynchronous delimited read operations	102
18	Sockets	104
18.1	Header <experimental/socket> synopsis	104
18.2	Requirements	106
18.3	Error codes	115
18.4	Class <code>socket_base</code>	116
18.5	Socket options	118
18.6	Class template <code>basic_socket</code>	121
18.7	Class template <code>basic_datagram_socket</code>	131
18.8	Class template <code>basic_stream_socket</code>	139
18.9	Class template <code>basic_socket_acceptor</code>	145
19	Socket iostreams	157
19.1	Class template <code>basic_socket_streambuf</code>	157
19.2	Class template <code>basic_socket_iostream</code>	161
20	Socket algorithms	164
20.1	Synchronous connect operations	164
20.2	Asynchronous connect operations	165
21	Internet protocol	167
21.1	Header <experimental/internet> synopsis	167
21.2	Requirements	171
21.3	Error codes	173
21.4	Class <code>ip::address</code>	174
21.5	Class <code>ip::address_v4</code>	177
21.6	Class <code>ip::address_v6</code>	181
21.7	Class <code>ip::bad_address_cast</code>	186
21.8	Hash support	187
21.9	Class template <code>ip::basic_address_iterator</code> specializations	187
21.10	Class template <code>ip::basic_address_range</code> specializations	188
21.11	Class template <code>ip::network_v4</code>	190
21.12	Class template <code>ip::network_v6</code>	193
21.13	Class template <code>ip::basic_endpoint</code>	195
21.14	Class template <code>ip::basic_resolver_entry</code>	199
21.15	Class template <code>ip::basic_resolver_results</code>	201
21.16	Class <code>ip::resolver_base</code>	204
21.17	Class template <code>ip::basic_resolver</code>	205
21.18	Host name functions	211
21.19	Class <code>ip::tcp</code>	211
21.20	Class <code>ip::udp</code>	212
21.21	Internet socket options	214
	Index	219
	Index of library names	221
	Index of implementation-defined behavior	227