

```
1  /**
2   * Copyright (c) 2011-2025 libbitcoin developers (see AUTHORS)
3   *
4   * This file is part of libbitcoin.
5   *
6   * This program is free software: you can redistribute it and/or modify
7   * it under the terms of the GNU Affero General Public License as published by
8   * the Free Software Foundation, either version 3 of the License, or
9   * (at your option) any later version.
10  *
11  * This program is distributed in the hope that it will be useful,
12  * but WITHOUT ANY WARRANTY; without even the implied warranty of
13  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14  * GNU Affero General Public License for more details.
15  *
16  * You should have received a copy of the GNU Affero General Public License
17  * along with this program. If not, see <http://www.gnu.org/licenses/>.
18  */
19 #ifndef LIBBITCOIN_NETWORK_DISTRIBUTORS_DISTRIBUTOR_PEER_HPP
20 #define LIBBITCOIN_NETWORK_DISTRIBUTORS_DISTRIBUTOR_PEER_HPP
21
22 #include <tuple>
23 #include <bitcoin/network/define.hpp>
24 #include <bitcoin/network/messages/peer/peer.hpp>
25 #include <bitcoin/network/messages/rpc/rpc.hpp>
26
27 namespace libbitcoin {
28     namespace network {
29         namespace rpc {
30
31             using namespace messages::peer;
32
33             struct peer_methods
34             {
35                 static constexpr std::tuple methods
```

```
36     {
37         method<"address", address::cptr>{"message"},
38         method<"alert", alert::cptr>{"message"},
39         method<"block", block::cptr>{"message"},
40         method<"bloom_filter_add", bloom_filter_add::cptr>{"message"},
41         method<"bloom_filter_clear", bloom_filter_clear::cptr>{"message"},
42         method<"bloom_filter_load", bloom_filter_load::cptr>{"message"},
43         method<"client_filter", client_filter::cptr>{"message"},
44         method<"client_filter_checkpoint", client_filter_checkpoint::cptr>{"message"},
45         method<"client_filter_headers", client_filter_headers::cptr>{"message"},
46         method<"compact_block", compact_block::cptr>{"message"},
47         method<"compact_transactions", compact_transactions::cptr>{"message"},
48         method<"fee_filter", fee_filter::cptr>{"message"},
49         method<"get_address", get_address::cptr>{"message"},
50         method<"get_blocks", get_blocks::cptr>{"message"},
51         method<"get_client_filter_checkpoint", get_client_filter_checkpoint::cptr>{"message"},
52         method<"get_client_filter_headers", get_client_filter_headers::cptr>{"message"},
53         method<"get_client_filters", get_client_filters::cptr>{"message"},
54         method<"get_compact_transactions", get_compact_transactions::cptr>{"message"},
55         method<"get_data", get_data::cptr>{"message"},
56         method<"get_headers", get_headers::cptr>{"message"},
57         method<"headers", headers::cptr>{"message"},
58         method<"inventory", inventory::cptr>{"message"},
59         method<"memory_pool", memory_pool::cptr>{"message"},
60         method<"merkle_block", merkle_block::cptr>{"message"},
61         method<"not_found", not_found::cptr>{"message"},
62         method<"ping", ping::cptr>{"message"},
63         method<"pong", pong::cptr>{"message"},
64         method<"reject", reject::cptr>{"message"},
65         method<"send_address_v2", send_address_v2::cptr>{"message"},
66         method<"send_compact", send_compact::cptr>{"message"},
67         method<"send_headers", send_headers::cptr>{"message"},
68         method<"transaction", transaction::cptr>{"message"},
69         method<"version", version::cptr>{"message"},
70         method<"version_acknowledge", version_acknowledge::cptr>{"message"},
```

```
71     method<"witness_tx_id_relay", witness_tx_id_relay::cptr>{"message"}
72 };
73
74     using address = at<0, decltype(methods)>;
75     using alert = at<1, decltype(methods)>;
76     using block = at<2, decltype(methods)>;
77     using bloom_filter_add = at<3, decltype(methods)>;
78     using bloom_filter_clear = at<4, decltype(methods)>;
79     using bloom_filter_load = at<5, decltype(methods)>;
80     using client_filter = at<6, decltype(methods)>;
81     using client_filter_checkpoint = at<7, decltype(methods)>;
82     using client_filter_headers = at<8, decltype(methods)>;
83     using compact_block = at<9, decltype(methods)>;
84     using compact_transactions = at<10, decltype(methods)>;
85     using fee_filter = at<11, decltype(methods)>;
86     using get_address = at<12, decltype(methods)>;
87     using get_blocks = at<13, decltype(methods)>;
88     using get_client_filter_checkpoint = at<14, decltype(methods)>;
89     using get_client_filter_headers = at<15, decltype(methods)>;
90     using get_client_filters = at<16, decltype(methods)>;
91     using get_compact_transactions = at<17, decltype(methods)>;
92     using get_data = at<18, decltype(methods)>;
93     using get_headers = at<19, decltype(methods)>;
94     using headers = at<20, decltype(methods)>;
95     using inventory = at<21, decltype(methods)>;
96     using memory_pool = at<22, decltype(methods)>;
97     using merkle_block = at<23, decltype(methods)>;
98     using not_found = at<24, decltype(methods)>;
99     using ping = at<25, decltype(methods)>;
100    using pong = at<26, decltype(methods)>;
101    using reject = at<27, decltype(methods)>;
102    using send_address_v2 = at<28, decltype(methods)>;
103    using send_compact = at<29, decltype(methods)>;
104    using send_headers = at<30, decltype(methods)>;
105    using transaction = at<31, decltype(methods)>;
```

```
106     using version = at<32, decltype(methods)>;
107     using version_acknowledge = at<33, decltype(methods)>;
108     using witness_tx_id_relay = at<34, decltype(methods)>;
109 };
110
111 } // namespace rpc
112 } // namespace network
113 } // namespace libbitcoin
114
115 #endif
116
```