YAODAQ

Yet An Other DAQ

Generated by Doxygen 1.9.3

1 License 1

2 Third-party licenses 3

2.1 LICENSE ISSUES . . . 4

2.1.1 OpenSSL License . . . 4

2.1.2 Original SSLeay License . . . 6

3 Namespace Index 11

3.1 Namespace List . . . 11

4 Hierarchical Index 13

4.1 Class Hierarchy . . . 13

5 Data Structure Index 15

5.1 Data Structures . . . 15

6 File Index 17

6.1 File List . . . 17

7 Namespace Documentation 19

7.1 spdlog Namespace Reference . . . 19

7.1.1 Detailed Description . . . 19

7.1.2 Typedef Documentation . . . 19

7.1.2.1 sink_ptr . . . 19

7.2 yaodaq Namespace Reference . . . 19

7.2.1 Detailed Description . . . 20

7.2.2 Enumeration Type Documentation . . . 20

7.2.2.1 Class . . . 20

7.2.2.2 Domain . . . 21

7.2.2.3 Family . . . 21

7.2.2.4 MessageType . . . 21

7.2.2.5 Severity . . . 22

7.2.2.6 Signal . . . 22

7.2.2.7 StatusCode . . . 23

7.2.3 Function Documentation . . . 23

7.2.3.1 operator<<() . . . 23

8 Data Structure Documentation 25

8.1 yaodaq::Close Class Reference . . . 25

8.1.1 Detailed Description . . . 25

8.1.2 Constructor & Destructor Documentation . . . 26

8.1.2.1 Close() [1/2] . . . 26

8.1.2.2 Close() [2/2] . . . 26

8.1.3 Member Function Documentation . . . 26

8.1.3.1 dump() . . . 26

8.1.3.2 get() . . . 26
8.1.3.3 getCode() . . . 26
8.1.3.4 getContent() . . . 26
8.1.3.5 getIdentifier() . . . 27
8.1.3.6 getReason() . . . 27
8.1.3.7 getRemote() . . . 27
8.1.3.8 getTime() . . . 27
8.1.3.9 getTimestamp() . . . 27
8.1.3.10 getTypeName() . . . 27
8.1.3.11 getTypeValue() . . . 27
8.1.3.12 setConnectionStateInfos() . . . 28
8.1.3.13 setContent() [1/3] . . . 28
8.1.3.14 setContent() [2/3] . . . 28
8.1.3.15 setContent() [3/3] . . . 28
8.1.3.16 setFrom() . . . 28
8.2 yaodaq::ConnectionState Class Reference . . . 28
8.2.1 Detailed Description . . . 29
8.2.2 Constructor & Destructor Documentation . . . 29
8.2.2.1 ConnectionState() . . . 29
8.2.2.2 ~ConnectionState() . . . 29
8.2.3 Member Function Documentation . . . 29
8.2.3.1 computeId() . . . 29
8.3 yaodaq::Error Class Reference . . . 30
8.3.1 Detailed Description . . . 30
8.3.2 Constructor & Destructor Documentation . . . 30
8.3.2.1 Error() [1/2] . . . 31
8.3.2.2 Error() [2/2] . . . 31
8.3.3 Member Function Documentation . . . 31
8.3.3.1 dump() . . . 31
8.3.3.2 get() . . . 31
8.3.3.3 getContent() . . . 31
8.3.3.4 getDecompressionError() . . . 31
8.3.3.5 getHttpStatus() . . . 32
8.3.3.6 getIdentifier() . . . 32
8.3.3.7 getReason() . . . 32
8.3.3.8 getRetries() . . . 32
8.3.3.9 getTime() . . . 32
8.3.3.10 getTimestamp() . . . 32
8.3.3.11 getTypeName() . . . 32
8.3.3.12 getTypeValue() . . . 33
8.3.3.13 getWaitTime() . . . 33
8.3.3.14 setConnectionStateInfos() . . . 33

8.3.3.15 setContent() [1/3] . . . 33
8.3.3.16 setContent() [2/3] . . . 33
8.3.3.17 setContent() [3/3] . . . 33
8.3.3.18 setFrom() . . . 33
8.4 yaodaq::Exception Class Reference . . . 34
8.4.1 Detailed Description . . . 34
8.4.2 Constructor & Destructor Documentation . . . 34
8.4.2.1 Exception() [1/2] . . . 34
8.4.2.2 Exception() [2/2] . . . 34
8.4.2.3 ~Exception() . . . 35
8.4.3 Member Function Documentation . . . 35
8.4.3.1 code() . . . 35
8.4.3.2 description() . . . 35
8.4.3.3 setFormat() . . . 35
8.4.3.4 setStyle() . . . 35
8.4.3.5 what() . . . 35
8.5 yaodaq::Fragment Class Reference . . . 35
8.5.1 Detailed Description . . . 36
8.5.2 Constructor & Destructor Documentation . . . 36
8.5.2.1 Fragment() [1/2] . . . 36
8.5.2.2 Fragment() [2/2] . . . 36
8.5.3 Member Function Documentation . . . 36
8.5.3.1 dump() . . . 36
8.5.3.2 get() . . . 37
8.5.3.3 getContent() . . . 37
8.5.3.4 getIdentifier() . . . 37
8.5.3.5 getTime() . . . 37
8.5.3.6 getTimestamp() . . . 37
8.5.3.7 getTypeName() . . . 37
8.5.3.8 getTypeValue() . . . 37
8.5.3.9 setConnectionStateInfos() . . . 38
8.5.3.10 setContent() [1/3] . . . 38
8.5.3.11 setContent() [2/3] . . . 38
8.5.3.12 setContent() [3/3] . . . 38
8.5.3.13 setFrom() . . . 38
8.6 yaodaq::Identifier Class Reference . . . 38
8.6.1 Detailed Description . . . 39
8.6.2 Constructor & Destructor Documentation . . . 39
8.6.2.1 Identifier() [1/2] . . . 39
8.6.2.2 Identifier() [2/2] . . . 39
8.6.3 Member Function Documentation . . . 39
8.6.3.1 empty() . . . 39

8.6.3.2 generateKey() . . . 40
8.6.3.3 get() . . . 40
8.6.3.4 getClass() . . . 40
8.6.3.5 getDomain() . . . 40
8.6.3.6 getFamily() . . . 40
8.6.3.7 getKey() . . . 40
8.6.3.8 getName() . . . 40
8.6.3.9 getType() . . . 40
8.6.3.10 operator<() . . . 41
8.6.3.11 parse() . . . 41
8.7 yaodaq::Interrupt Class Reference . . . 41
8.7.1 Detailed Description . . . 41
8.7.2 Constructor & Destructor Documentation . . . 41
8.7.2.1 Interrupt() . . . 42
8.7.2.2 ~Interrupt() . . . 42
8.7.3 Member Function Documentation . . . 42
8.7.3.1 getSignal() . . . 42
8.7.3.2 init() . . . 42
8.7.3.3 restore() . . . 42
8.8 yaodaq::IXMessage Class Reference . . . 42
8.8.1 Detailed Description . . . 43
8.8.2 Constructor & Destructor Documentation . . . 43
8.8.2.1 IXMessage() . . . 43
8.8.3 Member Function Documentation . . . 43
8.8.3.1 dump() . . . 43
8.8.3.2 get() . . . 44
8.8.3.3 getContent() . . . 44
8.8.3.4 getIdentifier() . . . 44
8.8.3.5 getTime() . . . 44
8.8.3.6 getTimestamp() . . . 44
8.8.3.7 getTypeName() . . . 44
8.8.3.8 getTypeValue() . . . 44
8.8.3.9 setConnectionStateInfos() . . . 45
8.8.3.10 setContent() [1/3] . . . 45
8.8.3.11 setContent() [2/3] . . . 45
8.8.3.12 setContent() [3/3] . . . 45
8.8.3.13 setFrom() . . . 45
8.9 yaodaq::Key Class Reference . . . 45
8.9.1 Detailed Description . . . 46
8.9.2 Constructor & Destructor Documentation . . . 46
8.9.2.1 Key() [1/2] . . . 46
8.9.2.2 Key() [2/2] . . . 46

8.9.3 Member Function Documentation . . . 46
8.9.3.1 getClass() . . . 46
8.9.3.2 getDomain() . . . 46
8.9.3.3 getFamily() . . . 46
8.9.3.4 getKey() . . . 47
8.10 yaodaq::LoggerHandler Class Reference . . . 47
8.10.1 Detailed Description . . . 47
8.10.2 Member Enumeration Documentation . . . 47
8.10.2.1 Verbosity . . . 47
8.10.3 Constructor & Destructor Documentation . . . 48
8.10.3.1 LoggerHandler() . . . 48
8.10.3.2 ∼LoggerHandler() . . . 48
8.10.4 Member Function Documentation . . . 48
8.10.4.1 addSink() . . . 48
8.10.4.2 clearSinks() . . . 48
8.10.4.3 logger() . . . 48
8.10.4.4 setName() . . . 48
8.10.4.5 setVerbosity() . . . 49
8.11 yaodaq::Looper Class Reference . . . 49
8.11.1 Detailed Description . . . 49
8.11.2 Constructor & Destructor Documentation . . . 49
8.11.2.1 Looper() . . . 49
8.11.2.2 ∼Looper() . . . 49
8.11.3 Member Function Documentation . . . 50
8.11.3.1 getInstance() . . . 50
8.11.3.2 getSignal() . . . 50
8.11.3.3 loop() . . . 50
8.11.3.4 supressInstance() . . . 50
8.12 yaodaq::Message Class Reference . . . 50
8.12.1 Detailed Description . . . 51
8.12.2 Constructor & Destructor Documentation . . . 51
8.12.2.1 Message() [1/5] . . . 51
8.12.2.2 Message() [2/5] . . . 51
8.12.2.3 Message() [3/5] . . . 52
8.12.2.4 Message() [4/5] . . . 52
8.12.2.5 Message() [5/5] . . . 52
8.12.3 Member Function Documentation . . . 52
8.12.3.1 dump() . . . 52
8.12.3.2 get() . . . 52
8.12.3.3 getContent() . . . 52
8.12.3.4 getIdentifier() . . . 52
8.12.3.5 getTime() . . . 53

8.12.3.6 getTimestamp() . . . 53
8.12.3.7 getTypeName() . . . 53
8.12.3.8 getTypeValue() . . . 53
8.12.3.9 setContent() [1/3] . . . 53
8.12.3.10 setContent() [2/3] . . . 53
8.12.3.11 setContent() [3/3] . . . 54
8.12.3.12 setFrom() . . . 54
8.13 yaodaq::Open Class Reference . . . 54
8.13.1 Detailed Description . . . 55
8.13.2 Constructor & Destructor Documentation . . . 55
8.13.2.1 Open() [1/2] . . . 55
8.13.2.2 Open() [2/2] . . . 55
8.13.3 Member Function Documentation . . . 55
8.13.3.1 dump() . . . 55
8.13.3.2 get() . . . 55
8.13.3.3 getContent() . . . 56
8.13.3.4 getHeaders() . . . 56
8.13.3.5 getIdentifier() . . . 56
8.13.3.6 getProtocol() . . . 56
8.13.3.7 getTime() . . . 56
8.13.3.8 getTimestamp() . . . 56
8.13.3.9 getTypeName() . . . 56
8.13.3.10 getTypeValue() . . . 57
8.13.3.11 getURI() . . . 57
8.13.3.12 setConnectionStateInfos() . . . 57
8.13.3.13 setContent() [1/3] . . . 57
8.13.3.14 setContent() [2/3] . . . 57
8.13.3.15 setContent() [3/3] . . . 57
8.13.3.16 setFrom() . . . 57
8.14 yaodaq::Ping Class Reference . . . 58
8.14.1 Detailed Description . . . 58
8.14.2 Constructor & Destructor Documentation . . . 58
8.14.2.1 Ping() [1/2] . . . 58
8.14.2.2 Ping() [2/2] . . . 59
8.14.3 Member Function Documentation . . . 59
8.14.3.1 dump() . . . 59
8.14.3.2 get() . . . 59
8.14.3.3 getContent() . . . 59
8.14.3.4 getIdentifier() . . . 59
8.14.3.5 getTime() . . . 59
8.14.3.6 getTimestamp() . . . 60
8.14.3.7 getTypeName() . . . 60

8.14.3.8 getTypeValue() . . . 60
8.14.3.9 setConnectionStateInfos() . . . 60
8.14.3.10 setContent() [1/3] . . . 60
8.14.3.11 setContent() [2/3] . . . 60
8.14.3.12 setContent() [3/3] . . . 60
8.14.3.13 setFrom() . . . 61
8.15 yaodaq::Pong Class Reference . . . 61
8.15.1 Detailed Description . . . 61
8.15.2 Constructor & Destructor Documentation . . . 62
8.15.2.1 Pong() [1/2] . . . 62
8.15.2.2 Pong() [2/2] . . . 62
8.15.3 Member Function Documentation . . . 62
8.15.3.1 dump() . . . 62
8.15.3.2 get() . . . 62
8.15.3.3 getContent() . . . 62
8.15.3.4 getIdentifier() . . . 62
8.15.3.5 getTime() . . . 63
8.15.3.6 getTimestamp() . . . 63
8.15.3.7 getTypeName() . . . 63
8.15.3.8 getTypeValue() . . . 63
8.15.3.9 setConnectionStateInfos() . . . 63
8.15.3.10 setContent() [1/3] . . . 63
8.15.3.11 setContent() [2/3] . . . 63
8.15.3.12 setContent() [3/3] . . . 64
8.15.3.13 setFrom() . . . 64
8.16 yaodaq::Version Class Reference . . . 64
8.16.1 Detailed Description . . . 64
8.16.2 Constructor & Destructor Documentation . . . 64
8.16.2.1 Version() [1/3] . . . 65
8.16.2.2 Version() [2/3] . . . 65
8.16.2.3 Version() [3/3] . . . 65
8.16.3 Member Function Documentation . . . 65
8.16.3.1 getMajor() . . . 65
8.16.3.2 getMinor() . . . 65
8.16.3.3 getPatch() . . . 65
8.16.3.4 getPreRelease() . . . 65
8.16.3.5 getPreReleaseNumber() . . . 65
8.17 yaodaq::WebsocketClient Class Reference . . . 66
8.17.1 Detailed Description . . . 66
8.17.2 Constructor & Destructor Documentation . . . 66
8.17.2.1 WebsocketClient() . . . 66
8.17.2.2 ~WebsocketClient() . . . 67

8.17.3 Member Function Documentation . . . 67
8.17.3.1 logger() . . . 67
8.17.3.2 loop() . . . 67
8.17.3.3 onClose() . . . 67
8.17.3.4 onError() . . . 67
8.17.3.5 onFragment() . . . 68
8.17.3.6 onMessage() . . . 68
8.17.3.7 onOpen() . . . 68
8.17.3.8 onPing() . . . 68
8.17.3.9 onPong() . . . 68
8.17.3.10 start() . . . 68
8.17.3.11 stop() . . . 68
8.18 yaodaq::WebsocketServer Class Reference . . . 69
8.18.1 Detailed Description . . . 69
8.18.2 Constructor & Destructor Documentation . . . 69
8.18.2.1 WebsocketServer() . . . 69
8.18.2.2 ~WebsocketServer() . . . 71
8.18.3 Member Function Documentation . . . 71
8.18.3.1 listen() . . . 71
8.18.3.2 logger() . . . 71
8.18.3.3 loop() . . . 71
8.18.3.4 onClose() . . . 71
8.18.3.5 onError() . . . 71
8.18.3.6 onFragment() . . . 72
8.18.3.7 onMessage() . . . 72
8.18.3.8 onOpen() . . . 72
8.18.3.9 onPing() . . . 72
8.18.3.10 onPong() . . . 72
8.18.3.11 sendToLoggers() [1/4] . . . 72
8.18.3.12 sendToLoggers() [2/4] . . . 72
8.18.3.13 sendToLoggers() [3/4] . . . 73
8.18.3.14 sendToLoggers() [4/4] . . . 73
8.18.3.15 setVerbosity() . . . 73
8.18.3.16 start() . . . 73
8.18.3.17 stop() . . . 73

9 File Documentation **75**

9.1 docs/License.md File Reference . . . 75
9.2 docs/Third-party licenses.md File Reference . . . 75
9.3 yaodaq/Classification.hpp File Reference . . . 75
9.4 Classification.hpp . . . 75
9.5 yaodaq/ConnectionState.hpp File Reference . . . 76

9.6 ConnectionState.hpp . . . 76
9.7 yaodaq/Exception.hpp File Reference . . . 77
9.8 Exception.hpp . . . 77
9.9 yaodaq/Identifier.hpp File Reference . . . 78
9.10 Identifier.hpp . . . 78
9.11 yaodaq/Interrupt.hpp File Reference . . . 78
9.12 Interrupt.hpp . . . 79
9.13 yaodaq/IXWebsocketMessage.hpp File Reference . . . 79
9.14 IXWebsocketMessage.hpp . . . 80
9.15 yaodaq/Key.hpp File Reference . . . 81
9.16 Key.hpp . . . 81
9.17 yaodaq/LoggerHandler.hpp File Reference . . . 81
9.18 LoggerHandler.hpp . . . 82
9.19 yaodaq/Looper.hpp File Reference . . . 82
9.20 Looper.hpp . . . 82
9.21 yaodaq/Message.hpp File Reference . . . 83
9.22 Message.hpp . . . 83
9.23 yaodaq/MessageType.hpp File Reference . . . 84
9.24 MessageType.hpp . . . 84
9.25 yaodaq/Severity.hpp File Reference . . . 85
9.26 Severity.hpp . . . 85
9.27 yaodaq/Signal.hpp File Reference . . . 85
9.28 Signal.hpp . . . 85
9.29 yaodaq/StatusCode.hpp File Reference . . . 86
9.30 StatusCode.hpp . . . 86
9.31 yaodaq/Version.hpp File Reference . . . 86
9.32 Version.hpp . . . 87
9.33 yaodaq/WebsocketClient.hpp File Reference . . . 87
9.34 WebsocketClient.hpp . . . 87
9.35 yaodaq/WebsocketServer.hpp File Reference . . . 88
9.36 WebsocketServer.hpp . . . 89
9.37 yaodaq/ConnectionState.cpp File Reference . . . 90
9.38 ConnectionState.cpp . . . 90
9.39 yaodaq/Exception.cpp File Reference . . . 90
9.40 Exception.cpp . . . 90
9.41 yaodaq/Identifier.cpp File Reference . . . 91
9.42 Identifier.cpp . . . 91
9.43 yaodaq/Interrupt.cpp File Reference . . . 92
9.44 Interrupt.cpp . . . 92
9.45 yaodaq/IXWebsocketMessage.cpp File Reference . . . 93
9.46 IXWebsocketMessage.cpp . . . 94
9.47 yaodaq/Key.cpp File Reference . . . 95

9.48 Key.cpp . . . 95
9.49 yaodaq/LoggerHandler.cpp File Reference . . . 95
9.50 LoggerHandler.cpp . . . 95
9.51 yaodaq/Looper.cpp File Reference . . . 96
9.52 Looper.cpp . . . 96
9.53 yaodaq/Message.cpp File Reference . . . 97
9.54 Message.cpp . . . 97
9.55 yaodaq/Version.cpp File Reference . . . 99
9.56 Version.cpp . . . 99
9.57 yaodaq/WebsocketClient.cpp File Reference . . . 99
9.58 WebsocketClient.cpp . . . 100
9.59 yaodaq/WebsocketServer.cpp File Reference . . . 101
9.60 WebsocketServer.cpp . . . 102

Chapter 1

License

Chapter 2

Third-party licenses

The following software may be included in this product: CPMLicenses. This software contains the following license and notice below:

MIT License

Copyright (c) 2021 Lars Melchior

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following software may be included in this product: magic_enum. This software contains the following license and notice below:

MIT License

Copyright (c) 2019 - 2021 Daniil Goncharov

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following software may be included in this product: zlib-ng. This software contains the following license and notice below:

(C) 1995-2013 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

The following software may be included in this product: OpenSSL-CMake. This software contains the following license and notice below:
MIT License
Copyright (c) 2020 flagarde
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
The following software may be included in this product: OpenSSL. This software contains the following license and notice below:

2.1 LICENSE ISSUES

The OpenSSL toolkit stays under a double license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts.

2.1.1 OpenSSL License

/* ==

- Copyright (c) 1998-2019 The OpenSSL Project. All rights reserved.
-
- Redistribution and use in source and binary forms, with or without
- modification, are permitted provided that the following conditions
- are met:
-
- 1. Redistributions of source code must retain the above copyright
- notice, this list of conditions and the following disclaimer.
-
- 2. Redistributions in binary form must reproduce the above copyright
- notice, this list of conditions and the following disclaimer in
- the documentation and/or other materials provided with the
- distribution.
-

- 3. All advertising materials mentioning features or use of this
- software must display the following acknowledgment:
- "This product includes software developed by the OpenSSL Project ∗ for use in the OpenSSL Toolkit. (http↩ ://www.openssl.org/)"
-
- 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
- endorse or promote products derived from this software without
- prior written permission. For written permission, please contact
- openssl-core@openssl.org.
-
- 5. Products derived from this software may not be called "OpenSSL"
- nor may "OpenSSL" appear in their names without prior written
- permission of the OpenSSL Project.
-
- 6. Redistributions of any form whatsoever must retain the following
- acknowledgment:
- "This product includes software developed by the OpenSSL Project ∗ for use in the OpenSSL Toolkit (http↩ ://www.openssl.org/)"
-
- THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
- EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
- IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
- PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
- ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
- SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
- NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
- LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
- HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
- STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
- ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
- OF THE POSSIBILITY OF SUCH DAMAGE.
- ==
-
- This product includes cryptographic software written by Eric Young
- (eay@cryptsoft.com). This product includes software written by Tim
- Hudson (tjh@cryptsoft.com).
- ∗/

2.1.2 Original SSLeay License

/∗ Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)

- All rights reserved.
-
- This package is an SSL implementation written
- by Eric Young (eay@cryptsoft.com).
- The implementation was written so as to conform with Netscapes SSL.
-
- This library is free for commercial and non-commercial use as long as
- the following conditions are aheared to. The following conditions
- apply to all code found in this distribution, be it the RC4, RSA,
- lhash, DES, etc., code; not just the SSL code. The SSL documentation
- included with this distribution is covered by the same copyright terms
- except that the holder is Tim Hudson (tjh@cryptsoft.com).
-
- Copyright remains Eric Young's, and as such any Copyright notices in
- the code are not to be removed.
- If this package is used in a product, Eric Young should be given attribution
- as the author of the parts of the library used.
- This can be in the form of a textual message at program startup or
- in documentation (online or textual) provided with the package.
-
- Redistribution and use in source and binary forms, with or without
- modification, are permitted provided that the following conditions
- are met:
- 1. Redistributions of source code must retain the copyright
- notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright
- notice, this list of conditions and the following disclaimer in the
- documentation and/or other materials provided with the distribution.
- 3. All advertising materials mentioning features or use of this software
- must display the following acknowledgement:
- "This product includes cryptographic software written by ∗ Eric Young (eay@cryptsoft.com)"
- The word 'cryptographic' can be left out if the rouines from the library
- being used are not cryptographic related :-).
- 4. If you include any Windows specific code (or a derivative thereof) from

- the apps directory (application code) you must include an acknowledgement:
- "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
-
- THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
- ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
- IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
- ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
- FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
- DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
- OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
- HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
- LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
- OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
- SUCH DAMAGE.
-
- The licence and distribution terms for any publically available version or
- derivative of this code cannot be changed. i.e. this code cannot simply be
- copied and put under another distribution licence
- [including the GNU Public Licence.] $*/$

The following software may be included in this product: IXWebSocket. This software contains the following license and notice below:
Copyright (c) 2018 Machine Zone, Inc. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The following software may be included in this product: fmt. This software contains the following license and notice below:
Copyright (c) 2012 - present, Victor Zverovich
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
— Optional exception to the license —
As an exception, if, as a result of your compiling your source code, portions of this Software are embedded into a machine-executable object form of such source code, you may redistribute such embedded portions in such object form without including the above copyright and permission notices.

The following software may be included in this product: spdlog. This software contains the following license and notice below:
The MIT License (MIT)
Copyright (c) 2016 Gabi Melman.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
– NOTE: Third party dependency used by this software – This software depends on the fmt lib (MIT License), and users must comply to its license: https://github.com/fmtlib/fmt/blob/master/LICENSE.↩
rst

The following software may be included in this product: nlohmann. This software contains the following license and notice below:
MIT License
Copyright (c) 2013-2022 Niels Lohmann
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following software may be included in this product: SourceLocation. This software contains the following license and notice below:
MIT License
Copyright (c) 2021 flagarde
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the

Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following software may be included in this product: Semver. This software contains the following license and notice below:
MIT License
Copyright (c) 2018 - 2021 Daniil Goncharov
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following software may be included in this product: CLI11. This software contains the following license and notice below:
CLI11 1.8 Copyright (c) 2017-2019 University of Cincinnati, developed by Henry Schreiner under NSF AWARD 1414736. All rights reserved.
Redistribution and use in source and binary forms of CLI11, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The following software may be included in this product: doctest. This software contains the following license and notice below:
The MIT License (MIT)
Copyright (c) 2016-2021 Viktor Kirilov
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE

LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

spdlog 19
yaodaq 19

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ix::ConnectionState
  yaodaq::ConnectionState 28
std::exception
  yaodaq::Exception 34
yaodaq::Identifier 38
yaodaq::Interrupt 41
yaodaq::Key 45
yaodaq::LoggerHandler 47
yaodaq::Looper 49
yaodaq::Message 50
  yaodaq::IXMessage 42
    yaodaq::Close 25
    yaodaq::Error 30
    yaodaq::Fragment 35
    yaodaq::Open 54
    yaodaq::Ping 58
    yaodaq::Pong 61
source_location
  yaodaq::Exception 34
semver::version
  yaodaq::Version 64
ix::WebSocket
  yaodaq::WebsocketClient 66
ix::WebSocketServer
  yaodaq::WebsocketServer 69

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

yaodaq::Close . . . 25
yaodaq::ConnectionState . . . 28
yaodaq::Error . . . 30
yaodaq::Exception . . . 34
yaodaq::Fragment . . . 35
yaodaq::Identifier . . . 38
yaodaq::Interrupt . . . 41
yaodaq::IXMessage . . . 42
yaodaq::Key . . . 45
yaodaq::LoggerHandler . . . 47
yaodaq::Looper . . . 49
yaodaq::Message . . . 50
yaodaq::Open . . . 54
yaodaq::Ping . . . 58
yaodaq::Pong . . . 61
yaodaq::Version . . . 64
yaodaq::WebsocketClient . . . 66
yaodaq::WebsocketServer . . . 69

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

yaodaq/Classification.hpp . . . 75
yaodaq/ConnectionState.hpp . . . 76
yaodaq/Exception.hpp . . . 77
yaodaq/Identifier.hpp . . . 78
yaodaq/Interrupt.hpp . . . 78
yaodaq/IXWebsocketMessage.hpp . . . 79
yaodaq/Key.hpp . . . 81
yaodaq/LoggerHandler.hpp . . . 81
yaodaq/Looper.hpp . . . 82
yaodaq/Message.hpp . . . 83
yaodaq/MessageType.hpp . . . 84
yaodaq/Severity.hpp . . . 85
yaodaq/Signal.hpp . . . 85
yaodaq/StatusCode.hpp . . . 86
yaodaq/Version.hpp . . . 86
yaodaq/WebsocketClient.hpp . . . 87
yaodaq/WebsocketServer.hpp . . . 88
yaodaq/ConnectionState.cpp . . . 90
yaodaq/Exception.cpp . . . 90
yaodaq/Identifier.cpp . . . 91
yaodaq/Interrupt.cpp . . . 92
yaodaq/IXWebsocketMessage.cpp . . . 93
yaodaq/Key.cpp . . . 95
yaodaq/LoggerHandler.cpp . . . 95
yaodaq/Looper.cpp . . . 96
yaodaq/Message.cpp . . . 97
yaodaq/Version.cpp . . . 99
yaodaq/WebsocketClient.cpp . . . 99
yaodaq/WebsocketServer.cpp . . . 101

Chapter 7

Namespace Documentation

7.1 spdlog Namespace Reference

Typedefs

- using sink_ptr = std::shared_ptr< spdlog::sinks::sink >

7.1.1 Detailed Description

Copyright

Copyright 2022 flagarde

7.1.2 Typedef Documentation

7.1.2.1 sink_ptr

```
using spdlog::sink_ptr = typedef std::shared_ptr<spdlog::sinks::sink>
```

Definition at line 15 of file LoggerHandler.hpp.

7.2 yaodaq Namespace Reference

Data Structures

- class Close
- class ConnectionState
- class Error
- class Exception
- class Fragment
- class Identifier
- class Interrupt
- class IXMessage
- class Key
- class LoggerHandler
- class Looper
- class Message
- class Open
- class Ping
- class Pong
- class Version
- class WebsocketClient
- class WebsocketServer

Enumerations

- enum class Domain : std::uint_least8_t { Unknown = 0 , Application = 1 , Web = 2 }
- enum class Class : std::uint_least8_t {
  Unknown = 0 , Server , Client , Module ,
  Board }
- enum class Family : std::uint_least16_t {
  Unknown = 0 , WebSocketServer , WebSocketClient , Logger ,
  Controller , Configurator , SlowController , Viewer ,
  Analyser , FileWriter }
- enum class MessageType : std::int_least16_t {
  Open = -1 , Close = -2 , Error = -3 , Ping = -4 ,
  Pong = -5 , Fragment = -6 , Unknown = 0 }
- enum class Severity : std::int_least16_t { Info = 1 , Warning = 10 , Error = 100 , Critical = 1000 }
- enum class Signal {
  NO = 0 , ABRT = static_cast<int>(Severity::Critical) + 1 , FPE = static_cast<int>(Severity::Critical) + 2 ,
  ILL = static_cast<int>(Severity::Critical) + 3 ,
  SEGV = static_cast<int>(Severity::Critical) + 4 , INT = static_cast<int>(Severity::Warning) + 1 , TERM =
  static_cast<int>(Severity::Warning) + 2 }
- enum class StatusCode : std::int_least32_t { SUCCESS = 0 , LISTEN_ERROR , WRONG_NUMBER_PARAMETERS
  , CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED = 4999 }

Functions

- std::ostream & operator<< (std::ostream &os, const MessageType &messageTypes)

7.2.1 Detailed Description

Copyright

Copyright 2022 flagarde

7.2.2 Enumeration Type Documentation

7.2.2.1 Class

```
enum class yaodaq::Class :  std::uint_least8_t  [strong]
```

Enumerator

Unknown	
Server	
Client	
Module	
Board	

Definition at line 22 of file Classification.hpp.

```
00023 {
00024   Unknown = 0,
00025   Server,
00026   Client,
00027   // Module is a client with start stop etc...
00028   Module,
00029   // Board is a module with a connector
00030   Board,
00031 };
```

7.2.2.2 Domain

```
enum class yaodaq::Domain :  std::uint_least8_t  [strong]
```

Enumerator

Unknown	
Application	
Web	

Definition at line 14 of file Classification.hpp.

```
00015 {
00016   Unknown     = 0,
00017   Application = 1,
00018   Web         = 2,
00019 };
```

7.2.2.3 Family

```
enum class yaodaq::Family :  std::uint_least16_t  [strong]
```

Enumerator

Unknown	
WebSocketServer	
WebSocketClient	
Logger	
Controller	
Configurator	
SlowController	
Viewer	
Analyser	
FileWriter	

Definition at line 34 of file Classification.hpp.

```
00035 {
00036   Unknown = 0,
00037   WebSocketServer,
00038   WebSocketClient,
00039   Logger,
00040   Controller,
00041   Configurator,
00042   SlowController,
00043   Viewer,
00044   Analyser,
00045   FileWriter,
00046 };
```

7.2.2.4 MessageType

```
enum class yaodaq::MessageType :  std::int_least16_t  [strong]
```

Enumerator

Open	
Close	
Error	
Ping	
Pong	
Fragment	
Unknown	

Definition at line 15 of file MessageType.hpp.

```
00016 {
00017   // IXWebSocket MessageType (Message is not set here)
00018   Open     = -1,
00019   Close    = -2,
00020   Error    = -3,
00021   Ping     = -4,
00022   Pong     = -5,
00023   Fragment = -6,
00024   // Unknown should not be used !
00025   Unknown  = 0,
00026 };
```

7.2.2.5 Severity

`enum class yaodaq::Severity : std::int_least16_t [strong]`

Enumerator

Info	
Warning	
Error	
Critical	

Definition at line 13 of file Severity.hpp.

```
00014 {
00015   Info     = 1,
00016   Warning  = 10,
00017   Error    = 100,
00018   Critical = 1000,
00019 };
```

7.2.2.6 Signal

`enum class yaodaq::Signal [strong]`

Enumerator

NO	
ABRT	
FPE	
ILL	
SEGV	
INT	
TERM	

Definition at line 15 of file Signal.hpp.

```
00016 {
00017   NO   = 0,  // No Signal.
00018   // Critical
00019   ABRT = static_cast<int>( Severity::Critical ) + 1,  // (Signal Abort) Abnormal termination, such as
        is initiated by the abort function.
00020   FPE  = static_cast<int>( Severity::Critical ) + 2,  // (Signal Floating-Point Exception) Erroneous
        arithmetic operation, such as zero divide or an operation resulting in overflow (not necessarily with
        a floating-point operation).
00021   ILL  = static_cast<int>( Severity::Critical ) + 3,  // (Signal Illegal Instruction) Invalid function
        image, such as an illegal instruction. This is generally due to a corruption in the code or to an
        attempt to execute data.
00022   SEGV = static_cast<int>( Severity::Critical ) + 4,  // (Signal Segmentation Violation) Invalid
        access to storage: When a program tries to read or write outside the memory it has allocated.
00023   // Warning
00024   INT  = static_cast<int>( Severity::Warning ) + 1,  // (Signal Interrupt) Interactive attention
        signal. Generally generated by the application user.
00025   TERM = static_cast<int>( Severity::Warning ) + 2,  // (Signal Terminate) Termination request sent to
        program.
00026 };
```

7.2.2.7 StatusCode

```
enum class yaodaq::StatusCode :  std::int_least32_t  [strong]
```

Enumerator

SUCCESS	
LISTEN_ERROR	
WRONG_NUMBER_PARAMETERS	
CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED	

Definition at line 13 of file StatusCode.hpp.

```
00014 {
00015   SUCCESS = 0,
00016   LISTEN_ERROR,
00017   WRONG_NUMBER_PARAMETERS,
00018   CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED = 4999,
00019 };
```

7.2.3 Function Documentation

7.2.3.1 operator<<()

```
std::ostream & yaodaq::operator<< (
            std::ostream & os,
            const MessageType & messageTypes )  [inline]
```

Definition at line 28 of file MessageType.hpp.

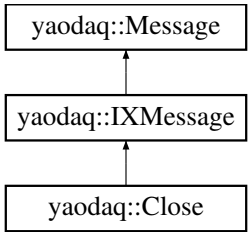
```
00028 { return os « static_cast<std::int_least8_t>( messageTypes ); }
```

Chapter 8

Data Structure Documentation

8.1 yaodaq::Close Class Reference

`#include <yaodaq/IXWebsocketMessage.hpp>`
Inheritance diagram for yaodaq::Close:



Public Member Functions

- Close (const ix::WebSocketCloseInfo &closeInfo)
- Close (const ix::WebSocketCloseInfo &closeInfo, std::shared_ptr< ConnectionState > &connectionState)
- std::uint16_t getCode () const
- std::string getReason () const
- bool getRemote () const
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- void setConnectionStateInfos (std::shared_ptr< ConnectionState > &connectionState)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.1.1 Detailed Description

Definition at line 41 of file IXWebsocketMessage.hpp.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 Close() [1/2]

```
yaodaq::Close::Close (
            const ix::WebSocketCloseInfo & closeInfo )  [explicit]
```

Definition at line 44 of file IXWebsocketMessage.cpp.

```
00044                                                   : IXMessage( MessageType::Close )
00045 {
00046   nlohmann::json j;
00047   j["code"]   = closeInfo.code;
00048   j["reason"] = closeInfo.reason;
00049   j["remote"] = closeInfo.remote;
00050   setContent( j );
00051 }
```

8.1.2.2 Close() [2/2]

```
yaodaq::Close::Close (
            const ix::WebSocketCloseInfo & closeInfo,
            std::shared_ptr< ConnectionState > & connectionState )
```

Definition at line 53 of file IXWebsocketMessage.cpp.

```
00053 : Close( closeInfo ) { setConnectionStateInfos( connectionState ); }
```

8.1.3 Member Function Documentation

8.1.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const  [inherited]
```

Definition at line 59 of file Message.cpp.

```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.1.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const  [inherited]
```

Definition at line 61 of file Message.cpp.

```
00061 { return m_JSON; }
```

8.1.3.3 getCode()

```
std::uint16_t yaodaq::Close::getCode ( ) const
```

Definition at line 55 of file IXWebsocketMessage.cpp.

```
00055 { return get()["content"]["code"].get<std::uint16_t>(); }
```

8.1.3.4 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const  [inherited]
```

Definition at line 67 of file Message.cpp.

```
00067 { return m_JSON["content"]; }
```

8.1.3.5 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const  [inherited]
```

Definition at line 89 of file Message.cpp.

```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
      m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
      ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
      ).value() );
00097     return id;
00098   }
00099 }
```

8.1.3.6 getReason()

```
std::string yaodaq::Close::getReason ( ) const
```

Definition at line 56 of file IXWebsocketMessage.cpp.

```
00056 { return get()["content"]["reason"].get<std::string>(); }
```

8.1.3.7 getRemote()

```
bool yaodaq::Close::getRemote ( ) const
```

Definition at line 57 of file IXWebsocketMessage.cpp.

```
00057 { return get()["content"]["remote"].get<bool>(); }
```

8.1.3.8 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const  [inherited]
```

Definition at line 71 of file Message.cpp.

```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
```

8.1.3.9 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const  [inherited]
```

Definition at line 69 of file Message.cpp.

```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.1.3.10 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const  [inherited]
```

Definition at line 63 of file Message.cpp.

```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.1.3.11 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const  [inherited]
```

Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.1.3.12 setConnectionStateInfos()

```
void yaodaq::IXMessage::setConnectionStateInfos (
            std::shared_ptr< ConnectionState > & connectionState )  [protected], [inherited]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
```

8.1.3.13 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected], [inherited]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.1.3.14 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected], [inherited]
```

Definition at line 39 of file Message.cpp.

```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.1.3.15 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected], [inherited]
```

Definition at line 41 of file Message.cpp.

```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

8.1.3.16 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )  [inherited]
```

Definition at line 80 of file Message.cpp.

```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```

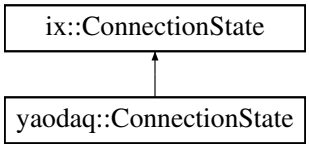
The documentation for this class was generated from the following files:

- yaodaq/IXWebsocketMessage.hpp
- yaodaq/IXWebsocketMessage.cpp

8.2 yaodaq::ConnectionState Class Reference

`#include <yaodaq/ConnectionState.hpp>`

Inheritance diagram for yaodaq::ConnectionState:



Public Member Functions

- virtual void computeId (const std::string &host, const Identifier &id) final
- ConnectionState ()
- virtual ~ConnectionState ()

8.2.1 Detailed Description

Definition at line 21 of file ConnectionState.hpp.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 ConnectionState()

```
yaodaq::ConnectionState::ConnectionState ( )
```

Definition at line 14 of file ConnectionState.cpp.

```
00014 : ix::ConnectionState() {}
```

8.2.2.2 ~ConnectionState()

```
yaodaq::ConnectionState::~ConnectionState ( )  [virtual]
```

Definition at line 16 of file ConnectionState.cpp.

```
00017 {
00018   std::lock_guard<std::mutex> guard( m_Mutex );
00019   m_Ids.remove( m_Pair );
00020 }
```

8.2.3 Member Function Documentation

8.2.3.1 computeId()

```
void yaodaq::ConnectionState::computeId (
            const std::string & host,
            const Identifier & id )  [final], [virtual]
```

Definition at line 22 of file ConnectionState.cpp.

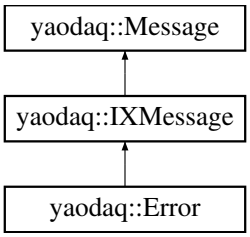
```
00023 {
00024   std::lock_guard<std::mutex> guard( m_Mutex );
00025   m_Pair = std::pair<std::string, std::string>( host, id.getName() );
00026
00027   if( id.empty() ) { _id = std::to_string( _globalId++ ); }
00028   else
00029   {
00030     std::list<std::pair<std::string, std::string>::iterator found = std::find( m_Ids.begin(),
      m_Ids.end(), m_Pair );
00031     if( found == m_Ids.end() )
00032     {
00033       _id = id.getName();
00034       m_Ids.push_back( m_Pair );
00035     }
00036     else
00037     {
00038       setTerminated();
00039     }
00040   }
00041 }
```

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

- yaodaq/ConnectionState.hpp
- yaodaq/ConnectionState.cpp

8.3 yaodaq::Error Class Reference

`#include <yaodaq/IXWebsocketMessage.hpp>`
Inheritance diagram for yaodaq::Error:



Public Member Functions

- Error (const ix::WebSocketErrorInfo &errorInfo)
- Error (const ix::WebSocketErrorInfo &errorInfo, std::shared_ptr< ConnectionState > &connectionState)
- std::uint16_t getRetries () const
- double getWaitTime () const
- int getHttpStatus () const
- std::string getReason () const
- bool getDecompressionError () const
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- void setConnectionStateInfos (std::shared_ptr< ConnectionState > &connectionState)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.3.1 Detailed Description

Definition at line 51 of file IXWebsocketMessage.hpp.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 Error() [1/2]

```
yaodaq::Error::Error (
            const ix::WebSocketErrorInfo & errorInfo )  [explicit]
```

Definition at line 60 of file IXWebsocketMessage.cpp.

```
00060                                                 : IXMessage( MessageType::Error )
00061 {
00062   nlohmann::json j;
00063   j["retries"]             = errorInfo.retries;
00064   j["wait_time"]           = errorInfo.wait_time;
00065   j["http_status"]         = errorInfo.http_status;
00066   j["reason"]              = errorInfo.reason;
00067   j["decompression_error"] = errorInfo.decompressionError;
00068   setContent( j );
00069 }
```

8.3.2.2 Error() [2/2]

```
yaodaq::Error::Error (
            const ix::WebSocketErrorInfo & errorInfo,
            std::shared_ptr< ConnectionState > & connectionState )
```

Definition at line 71 of file IXWebsocketMessage.cpp.

```
00071 : Error( errorInfo ) { setConnectionStateInfos( connectionState ); }
```

8.3.3 Member Function Documentation

8.3.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const  [inherited]
```

Definition at line 59 of file Message.cpp.

```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.3.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const  [inherited]
```

Definition at line 61 of file Message.cpp.

```
00061 { return m_JSON; }
```

8.3.3.3 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const  [inherited]
```

Definition at line 67 of file Message.cpp.

```
00067 { return m_JSON["content"]; }
```

8.3.3.4 getDecompressionError()

```
bool yaodaq::Error::getDecompressionError ( ) const
```

Definition at line 81 of file IXWebsocketMessage.cpp.

```
00081 { return get()["content"]["decompression_error"].get<bool>(); }
```

8.3.3.5 getHttpStatus()

```
int yaodaq::Error::getHttpStatus ( ) const
```

Definition at line 77 of file IXWebsocketMessage.cpp.

```
00077 { return get()["content"]["http_status"].get<int>(); }
```

8.3.3.6 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const  [inherited]
```

Definition at line 89 of file Message.cpp.

```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
       ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00097     return id;
00098   }
00099 }
```

8.3.3.7 getReason()

```
std::string yaodaq::Error::getReason ( ) const
```

Definition at line 79 of file IXWebsocketMessage.cpp.

```
00079 { return get()["content"]["reason"].get<std::string>(); }
```

8.3.3.8 getRetries()

```
std::uint16_t yaodaq::Error::getRetries ( ) const
```

Definition at line 73 of file IXWebsocketMessage.cpp.

```
00073 { return get()["content"]["retries"].get<std::uint16_t>(); }
```

8.3.3.9 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const  [inherited]
```

Definition at line 71 of file Message.cpp.

```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
```

8.3.3.10 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const  [inherited]
```

Definition at line 69 of file Message.cpp.

```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.3.3.11 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const  [inherited]
```

Definition at line 63 of file Message.cpp.

```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.3.3.12 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const  [inherited]
```

Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.3.3.13 getWaitTime()

```
double yaodaq::Error::getWaitTime ( ) const
```

Definition at line 75 of file IXWebsocketMessage.cpp.

```
00075 { return get()["content"]["wait_time"].get<double>(); }
```

8.3.3.14 setConnectionStateInfos()

```
void yaodaq::IXMessage::setConnectionStateInfos (
            std::shared_ptr< ConnectionState > & connectionState )  [protected], [inherited]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
```

8.3.3.15 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected], [inherited]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.3.3.16 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected], [inherited]
```

Definition at line 39 of file Message.cpp.

```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.3.3.17 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected], [inherited]
```

Definition at line 41 of file Message.cpp.

```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

8.3.3.18 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )  [inherited]
```

Definition at line 80 of file Message.cpp.

```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
```

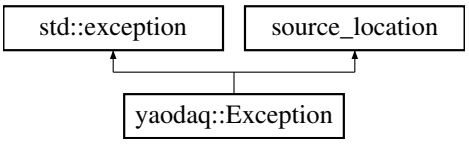
```
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```

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

- yaodaq/IXWebsocketMessage.hpp
- yaodaq/IXWebsocketMessage.cpp

8.4 yaodaq::Exception Class Reference

`#include <yaodaq/Exception.hpp>`
Inheritance diagram for yaodaq::Exception:



Public Member Functions

- Exception ()=delete
- Exception (const StatusCode &statusCode, const std::string &description, const source_location &location=source_location::current())
- ~Exception () noexcept override=default
- const char * what () const noexcept final
- const char * description () const noexcept
- std::int_least32_t code () const noexcept

Static Public Member Functions

- static void setFormat (const std::string &format)
- static void setStyle (const fmt::text_style &style={})

8.4.1 Detailed Description

Definition at line 19 of file Exception.hpp.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 Exception() [1/2]

```
yaodaq::Exception::Exception ( )  [delete]
```

8.4.2.2 Exception() [2/2]

```
yaodaq::Exception::Exception (
            const StatusCode & statusCode,
            const std::string & description,
            const source_location & location = source_location::current() )
```

Definition at line 14 of file Exception.cpp.

```
00014 : source_location( location ), m_Code( static_cast<std::int_least32_t>( statusCode ) ), m_Description(
      description ) { constructMessage(); }
```

8.4.2.3 ~Exception()

```
yaodaq::Exception::~Exception ( )  [override], [default], [noexcept]
```

8.4.3 Member Function Documentation

8.4.3.1 code()

```
std::int_least32_t yaodaq::Exception::code ( ) const  [noexcept]
```

Definition at line 20 of file Exception.cpp.

```
00020 { return m_Code; }
```

8.4.3.2 description()

```
const char * yaodaq::Exception::description ( ) const  [noexcept]
```

Definition at line 18 of file Exception.cpp.

```
00018 { return m_Description.c_str(); }
```

8.4.3.3 setFormat()

```
static void yaodaq::Exception::setFormat (
            const std::string & format )  [inline], [static]
```

Definition at line 24 of file Exception.hpp.

```
00024 { m_Format = format; }
```

8.4.3.4 setStyle()

```
static void yaodaq::Exception::setStyle (
            const fmt::text_style & style = {} )  [inline], [static]
```

Definition at line 26 of file Exception.hpp.

```
00026 {} ) { m_Style = style; }
```

8.4.3.5 what()

```
const char * yaodaq::Exception::what ( ) const  [final], [noexcept]
```

Definition at line 16 of file Exception.cpp.

```
00016 { return m_Message.c_str(); }
```

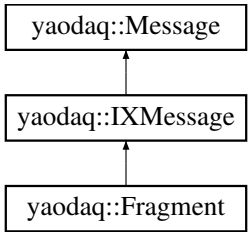
The documentation for this class was generated from the following files:

- yaodaq/Exception.hpp
- yaodaq/Exception.cpp

8.5 yaodaq::Fragment Class Reference

```
#include <yaodaq/IXWebsocketMessage.hpp>
```

Inheritance diagram for yaodaq::Fragment:



Public Member Functions

- Fragment (const ix::WebSocketMessagePtr &fragment)
- Fragment (const ix::WebSocketMessagePtr &fragment, std::shared_ptr< ConnectionState > &connection↩ State)
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- void setConnectionStateInfos (std::shared_ptr< ConnectionState > &connectionState)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.5.1 Detailed Description

Definition at line 77 of file IXWebsocketMessage.hpp.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Fragment() [1/2]

```
yaodaq::Fragment::Fragment (
            const ix::WebSocketMessagePtr & fragment )  [explicit]
```

Definition at line 94 of file IXWebsocketMessage.cpp.

```
00094 : IXMessage( MessageType::Fragment ) {}
```

8.5.2.2 Fragment() [2/2]

```
yaodaq::Fragment::Fragment (
            const ix::WebSocketMessagePtr & fragment,
            std::shared_ptr< ConnectionState > & connectionState )
```

Definition at line 96 of file IXWebsocketMessage.cpp.

```
00096 : Fragment( fragment ) { setConnectionStateInfos( connectionState ); }
```

8.5.3 Member Function Documentation

8.5.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const  [inherited]
```

Definition at line 59 of file Message.cpp.
```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.5.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const  [inherited]
```
Definition at line 61 of file Message.cpp.
```
00061 { return m_JSON; }
```

8.5.3.3 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const  [inherited]
```
Definition at line 67 of file Message.cpp.
```
00067 { return m_JSON["content"]; }
```

8.5.3.4 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const  [inherited]
```
Definition at line 89 of file Message.cpp.
```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
       ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00097     return id;
00098   }
00099 }
```

8.5.3.5 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const  [inherited]
```
Definition at line 71 of file Message.cpp.
```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
```

8.5.3.6 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const  [inherited]
```
Definition at line 69 of file Message.cpp.
```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.5.3.7 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const  [inherited]
```
Definition at line 63 of file Message.cpp.
```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.5.3.8 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const  [inherited]
```
Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.5.3.9 setConnectionStateInfos()

```
void yaodaq::IXMessage::setConnectionStateInfos (
            std::shared_ptr< ConnectionState > & connectionState )  [protected], [inherited]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
```

8.5.3.10 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected], [inherited]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.5.3.11 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected], [inherited]
```

Definition at line 39 of file Message.cpp.

```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.5.3.12 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected], [inherited]
```

Definition at line 41 of file Message.cpp.

```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

8.5.3.13 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )  [inherited]
```

Definition at line 80 of file Message.cpp.

```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```

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

- yaodaq/IXWebsocketMessage.hpp
- yaodaq/IXWebsocketMessage.cpp

8.6 yaodaq::Identifier Class Reference

```
#include <yaodaq/Identifier.hpp>
```

Public Member Functions

- Identifier ()=default
- Identifier (const std::string &type, const std::string &name)
- void generateKey (const Domain &domain=Domain::Unknown, const Class &c_lass=Class::Unknown, const Family &family=Family::Unknown)
- std::string getDomain () const
- std::string getClass () const
- std::string getFamily () const
- std::string getType () const
- std::string getName () const
- Key getKey () const
- std::string get () const
- bool empty () const
- bool operator< (const Identifier &) const

Static Public Member Functions

- static Identifier parse (const std::string &)

8.6.1 Detailed Description

Definition at line 16 of file Identifier.hpp.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 Identifier() [1/2]

```
yaodaq::Identifier::Identifier ( )  [default]
```

8.6.2.2 Identifier() [2/2]

```
yaodaq::Identifier::Identifier (
            const std::string & type,
            const std::string & name )
```

Definition at line 26 of file Identifier.cpp.

```
00026 : m_Type( type ), m_Name( name ) {}
```

8.6.3 Member Function Documentation

8.6.3.1 empty()

```
bool yaodaq::Identifier::empty ( ) const
```

Definition at line 19 of file Identifier.cpp.

```
00020 {
00021   if( get() == Identifier().get() ) return true;
00022   else
00023     return false;
00024 }
```

8.6.3.2 generateKey()

```
void yaodaq::Identifier::generateKey (
            const Domain & domain = Domain::Unknown,
            const Class & c_lass = Class::Unknown,
            const Family & family = Family::Unknown )
```

Definition at line 28 of file Identifier.cpp.

```
00028 { m_Key = Key( domain, c_lass, family ); }
```

8.6.3.3 get()

```
std::string yaodaq::Identifier::get ( ) const
```

Definition at line 42 of file Identifier.cpp.

```
00042 { return fmt::format( "{0}/{1}/{2}/{3}/{4}", getDomain(), getClass(), getFamily(), getType(),
       getName() ); }
```

8.6.3.4 getClass()

```
std::string yaodaq::Identifier::getClass ( ) const
```

Definition at line 32 of file Identifier.cpp.

```
00032 { return static_cast<std::string>( magic_enum::enum_name( magic_enum::enum_cast<Class>(
       m_Key.getClass() ).value() ) ); }
```

8.6.3.5 getDomain()

```
std::string yaodaq::Identifier::getDomain ( ) const
```

Definition at line 30 of file Identifier.cpp.

```
00030 { return static_cast<std::string>( magic_enum::enum_name( magic_enum::enum_cast<Domain>(
       m_Key.getDomain() ).value() ) ); }
```

8.6.3.6 getFamily()

```
std::string yaodaq::Identifier::getFamily ( ) const
```

Definition at line 34 of file Identifier.cpp.

```
00034 { return static_cast<std::string>( magic_enum::enum_name( magic_enum::enum_cast<Family>(
       m_Key.getFamily() ).value() ) ); }
```

8.6.3.7 getKey()

```
Key yaodaq::Identifier::getKey ( ) const
```

Definition at line 40 of file Identifier.cpp.

```
00040 { return m_Key; }
```

8.6.3.8 getName()

```
std::string yaodaq::Identifier::getName ( ) const
```

Definition at line 38 of file Identifier.cpp.

```
00038 { return m_Name; }
```

8.6.3.9 getType()

```
std::string yaodaq::Identifier::getType ( ) const
```

Definition at line 36 of file Identifier.cpp.

```
00036 { return m_Type; }
```

8.6.3.10 operator<()

```
bool yaodaq::Identifier::operator< (
            const Identifier & identifier ) const
```

Definition at line 75 of file Identifier.cpp.

```
00075 { return this->get() < identifier.get(); }
```

8.6.3.11 parse()

```
Identifier yaodaq::Identifier::parse (
            const std::string & id )  [static]
```

Definition at line 44 of file Identifier.cpp.

```
00045 {
00046   std::vector<std::string> result;
00047   std::string              tmp        = id;
00048   std::string              separator  = "/";
00049   std::size_t              second_pos = tmp.find( separator );
00050   while( second_pos != std::string::npos )
00051   {
00052     if( 0 != second_pos )
00053     {
00054       std::string word = tmp.substr( 0, second_pos - 0 );
00055       result.push_back( word );
00056     }
00057     else
00058       result.push_back( "" );
00059     tmp        = tmp.substr( second_pos + separator.length() );
00060     second_pos = tmp.find( separator );
00061     if( second_pos == std::string::npos ) result.push_back( tmp );
00062   }
00063   if( result.size() == 5 )
00064   {
00065     Identifier identifier( result[3], result[4] );
00066     identifier.generateKey( magic_enum::enum_cast<Domain>( result[0] ).value(),
      magic_enum::enum_cast<Class>( result[1] ).value(), magic_enum::enum_cast<Family>( result[2] ).value()
      );
00067     return identifier;
00068   }
00069   else
00070   {
00071     throw Exception( StatusCode::WRONG_NUMBER_PARAMETERS, "Number of parameters in key should be 5
     (Domain/Class/Family/Type/Name) !" );
00072   }
00073 }
```

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

- yaodaq/Identifier.hpp
- yaodaq/Identifier.cpp

8.7 yaodaq::Interrupt Class Reference

`#include <yaodaq/Interrupt.hpp>`

Public Member Functions

- Interrupt ()
- void init ()
- void restore ()
- Signal getSignal ()
- ~Interrupt ()

8.7.1 Detailed Description

Definition at line 19 of file Interrupt.hpp.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 Interrupt()

```
yaodaq::Interrupt::Interrupt ( )
```
Definition at line 19 of file Interrupt.cpp.
```
00019 { init(); }
```

8.7.2.2 ~Interrupt()

```
yaodaq::Interrupt::~Interrupt ( )
```
Definition at line 42 of file Interrupt.cpp.
```
00042 { restore(); }
```

8.7.3 Member Function Documentation

8.7.3.1 getSignal()

```
Signal yaodaq::Interrupt::getSignal ( )
```
Definition at line 44 of file Interrupt.cpp.
```
00045 {
00046   if( m_Signal.load() != Signal::NO )
00047   {
00048     std::lock_guard<std::mutex> guard( m_mutex );
00049     init();
00050   }
00051   return m_Signal.load();
00052 }
```

8.7.3.2 init()

```
void yaodaq::Interrupt::init ( )
```
Definition at line 31 of file Interrupt.cpp.
```
00032 {
00033   setSignal( Signal::TERM );
00034   setSignal( Signal::TERM );
00035   setSignal( Signal::SEGV );
00036   setSignal( Signal::INT );
00037   setSignal( Signal::ILL );
00038   setSignal( Signal::ABRT );
00039   setSignal( Signal::FPE );
00040 }
```

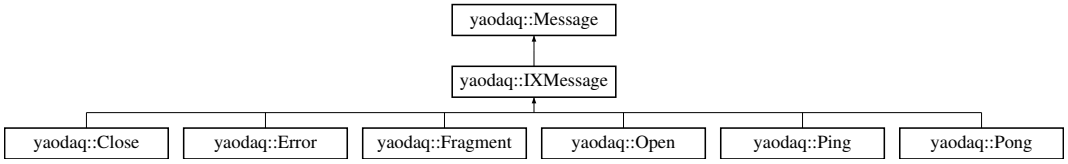
8.7.3.3 restore()

```
void yaodaq::Interrupt::restore ( )
```
Definition at line 21 of file Interrupt.cpp.
```
00022 {
00023   std::signal( SIGTERM, SIG_DFL );
00024   std::signal( SIGSEGV, SIG_DFL );
00025   std::signal( SIGINT, SIG_DFL );
00026   std::signal( SIGILL, SIG_DFL );
00027   std::signal( SIGABRT, SIG_DFL );
00028   std::signal( SIGFPE, SIG_DFL );
00029 }
```
The documentation for this class was generated from the following files:

- yaodaq/Interrupt.hpp
- yaodaq/Interrupt.cpp

8.8 yaodaq::IXMessage Class Reference

```
#include <yaodaq/IXWebsocketMessage.hpp>
```
Inheritance diagram for yaodaq::IXMessage:



Public Member Functions

- IXMessage (const MessageType &messageType)
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- void setConnectionStateInfos (std::shared_ptr< ConnectionState > &connectionState)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.8.1 Detailed Description

Definition at line 22 of file IXWebsocketMessage.hpp.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 IXMessage()

```
yaodaq::IXMessage::IXMessage (
            const MessageType & messageType )  [explicit]
```

Definition at line 10 of file IXWebsocketMessage.cpp.

```
00010 : Message( messageType ) {}
```

8.8.3 Member Function Documentation

8.8.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const  [inherited]
```

Definition at line 59 of file Message.cpp.

```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.8.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const  [inherited]
```

Definition at line 61 of file Message.cpp.

```
00061 { return m_JSON; }
```

8.8.3.3 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const  [inherited]
```

Definition at line 67 of file Message.cpp.

```
00067 { return m_JSON["content"]; }
```

8.8.3.4 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const  [inherited]
```

Definition at line 89 of file Message.cpp.

```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
       ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00097     return id;
00098   }
00099 }
```

8.8.3.5 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const  [inherited]
```

Definition at line 71 of file Message.cpp.

```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
```

8.8.3.6 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const  [inherited]
```

Definition at line 69 of file Message.cpp.

```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.8.3.7 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const  [inherited]
```

Definition at line 63 of file Message.cpp.

```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.8.3.8 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const  [inherited]
```

Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.8.3.9 setConnectionStateInfos()

```
void yaodaq::IXMessage::setConnectionStateInfos (
            std::shared_ptr< ConnectionState > & connectionState )  [protected]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
```

8.8.3.10 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected], [inherited]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.8.3.11 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected], [inherited]
```

Definition at line 39 of file Message.cpp.

```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.8.3.12 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected], [inherited]
```

Definition at line 41 of file Message.cpp.

```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

8.8.3.13 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )  [inherited]
```

Definition at line 80 of file Message.cpp.

```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```

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

- yaodaq/IXWebsocketMessage.hpp
- yaodaq/IXWebsocketMessage.cpp

8.9 yaodaq::Key Class Reference

```
#include <yaodaq/Key.hpp>
```

Public Member Functions

- Key ()=default
- Key (const Domain &domain, const Class &c_lass, const Family &family)
- std::int_least8_t getDomain () const
- std::int_least8_t getClass () const
- std::int_least16_t getFamily () const
- std::int_least32_t getKey () const

8.9.1 Detailed Description

Definition at line 15 of file Key.hpp.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 Key() [1/2]

```
yaodaq::Key::Key ( )  [default]
```

8.9.2.2 Key() [2/2]

```
yaodaq::Key::Key (
            const Domain & domain,
            const Class & c_lass,
            const Family & family )  [explicit]
```

Definition at line 11 of file Key.cpp.

```
00011 { m_Key = ( static_cast<std::int_least8_t>( domain ) « 24 ) + ( static_cast<std::int_least8_t>( c_lass
      ) « 16 ) + static_cast<std::int_least16_t>( family ); }
```

8.9.3 Member Function Documentation

8.9.3.1 getClass()

```
std::int_least8_t yaodaq::Key::getClass ( ) const
```

Definition at line 15 of file Key.cpp.

```
00015 { return ( m_Key » 16 ) & 0xFF; }
```

8.9.3.2 getDomain()

```
std::int_least8_t yaodaq::Key::getDomain ( ) const
```

Definition at line 13 of file Key.cpp.

```
00013 { return ( m_Key » 24 ) & 0xFF; }
```

8.9.3.3 getFamily()

```
std::int_least16_t yaodaq::Key::getFamily ( ) const
```

Definition at line 17 of file Key.cpp.

```
00017 { return (m_Key)&0xFFFF; }
```

8.9.3.4 getKey()

```
std::int_least32_t yaodaq::Key::getKey ( ) const
```

Definition at line 19 of file Key.cpp.

```
00019 { return m_Key; }
```

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

- yaodaq/Key.hpp
- yaodaq/Key.cpp

8.10 yaodaq::LoggerHandler Class Reference

```
#include <yaodaq/LoggerHandler.hpp>
```

Public Types

- enum class Verbosity {
  Off , Trace , Debug , Info ,
  Warn , Error , Critical }

Public Member Functions

- LoggerHandler ()
- ~LoggerHandler ()
- void setVerbosity (const Verbosity &verbosity)
- void setName (const std::string &)
- std::shared_ptr< spdlog::logger > logger ()
- void addSink (const spdlog::sink_ptr &)
- void clearSinks ()

8.10.1 Detailed Description

Definition at line 21 of file LoggerHandler.hpp.

8.10.2 Member Enumeration Documentation

8.10.2.1 Verbosity

```
enum class yaodaq::LoggerHandler::Verbosity  [strong]
```

Enumerator

Off	
Trace	
Debug	
Info	
Warn	
Error	
Critical	

Definition at line 24 of file LoggerHandler.hpp.

```
00025   {
00026     Off,
00027     Trace,
00028     Debug,
00029     Info,
00030     Warn,
00031     Error,
```

```
00032     Critical
00033   };
```

8.10.3 Constructor & Destructor Documentation

8.10.3.1 LoggerHandler()

```
yaodaq::LoggerHandler::LoggerHandler ( )
```

Definition at line 12 of file LoggerHandler.cpp.

```
00012 { init(); }
```

8.10.3.2 ~LoggerHandler()

```
yaodaq::LoggerHandler::~LoggerHandler ( )
```

Definition at line 20 of file LoggerHandler.cpp.

```
00020 {}
```

8.10.4 Member Function Documentation

8.10.4.1 addSink()

```
void yaodaq::LoggerHandler::addSink (
              const spdlog::sink_ptr & sink )
```

Definition at line 45 of file LoggerHandler.cpp.

```
00046 {
00047   m_Sinks.push_back( sink );
00048   init();
00049 }
```

8.10.4.2 clearSinks()

```
void yaodaq::LoggerHandler::clearSinks ( )
```

Definition at line 51 of file LoggerHandler.cpp.

```
00052 {
00053   m_Sinks.clear();
00054   init();
00055 }
```

8.10.4.3 logger()

```
std::shared_ptr< spdlog::logger > yaodaq::LoggerHandler::logger ( )
```

Definition at line 43 of file LoggerHandler.cpp.

```
00043 { return std::shared_ptr<spdlog::logger>( m_Logger ); }
```

8.10.4.4 setName()

```
void yaodaq::LoggerHandler::setName (
              const std::string & name )
```

Definition at line 14 of file LoggerHandler.cpp.

```
00015 {
00016   m_Name = name;
00017   init();
00018 }
```

8.10.4.5 setVerbosity()

```
void yaodaq::LoggerHandler::setVerbosity (
            const Verbosity & verbosity )
```

Definition at line 22 of file LoggerHandler.cpp.

```
00023 {
00024   m_Verbosity = verbosity;
00025   init();
00026 }
```

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

- yaodaq/LoggerHandler.hpp
- yaodaq/LoggerHandler.cpp

8.11 yaodaq::Looper Class Reference

`#include <yaodaq/Looper.hpp>`

Public Member Functions

- Looper ()
- Signal loop ()
- Signal getSignal ()
- void supressInstance ()
- ~Looper ()

Static Public Member Functions

- static int getInstance ()

8.11.1 Detailed Description

Definition at line 15 of file Looper.hpp.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 Looper()

```
yaodaq::Looper::Looper ( )
```

Definition at line 28 of file Looper.cpp.

```
00029 {
00030   if( m_hasBeenAdded == false )
00031   {
00032     m_hasBeenAdded = true;
00033     ++m_instance;
00034   }
00035 }
```

8.11.2.2 ~Looper()

```
yaodaq::Looper::~Looper ( )
```

Definition at line 52 of file Looper.cpp.

```
00053 {
00054   if( m_hasBeenAdded == true && m_hasBeenSupressed == false )
00055   {
00056     m_hasBeenSupressed = true;
00057     --m_instance;
00058   }
00059 }
```

8.11.3 Member Function Documentation

8.11.3.1 getInstance()

```
int yaodaq::Looper::getInstance ( )  [static]
```

Definition at line 17 of file Looper.cpp.

```
00017 { return m_instance; }
```

8.11.3.2 getSignal()

```
Signal yaodaq::Looper::getSignal ( )
```

Definition at line 50 of file Looper.cpp.

```
00050 { return m_Interrupt.getSignal(); }
```

8.11.3.3 loop()

```
Signal yaodaq::Looper::loop ( )
```

Definition at line 37 of file Looper.cpp.

```
00038 {
00039   static Signal signal{ yaodaq::Signal::NO };
00040   if( m_instance == 0 )
00041   {
00042     do {
00043       signal = m_Interrupt.getSignal();
00044       std::this_thread::sleep_for( std::chrono::microseconds( 1 ) );
00045     } while( signal == yaodaq::Signal::NO );
00046   }
00047   return signal;
00048 }
```

8.11.3.4 supressInstance()

```
void yaodaq::Looper::supressInstance ( )
```

Definition at line 19 of file Looper.cpp.

```
00020 {
00021   if( m_hasBeenSupressed == false )
00022   {
00023     m_hasBeenSupressed = true;
00024     m_instance--;
00025   }
00026 }
```

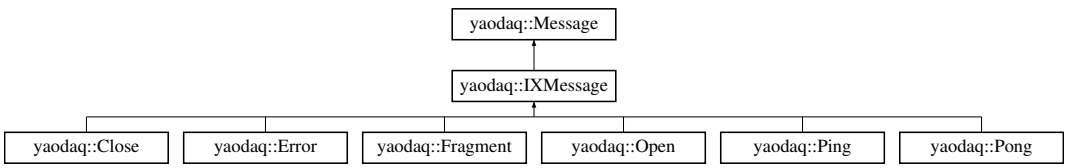
The documentation for this class was generated from the following files:

- yaodaq/Looper.hpp
- yaodaq/Looper.cpp

8.12 yaodaq::Message Class Reference

```
#include <yaodaq/Message.hpp>
```

Inheritance diagram for yaodaq::Message:



Public Member Functions

- Message ()
- Message (const nlohmann::json &content, const MessageType &messageType=MessageType::Unknown)
- Message (const std::string &content, const MessageType &messageType=MessageType::Unknown)
- Message (const char *content, const MessageType &messageType=MessageType::Unknown)
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- Message (const MessageType &messageType)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.12.1 Detailed Description

Definition at line 18 of file Message.hpp.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 Message() [1/5]

```
yaodaq::Message::Message ( )
```

Definition at line 24 of file Message.cpp.

```
00025 {
00026   m_JSON["from"];
00027   m_JSON["to"];
00028   m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
00029   // m_JSON["uuid"] = ix::uuid4();
00030   m_JSON["content"];
00031   m_JSON["timestamp"] = fmt::format( "{:%F %T %z}", fmt::gmtime( std::chrono::system_clock::to_time_t(
       std::chrono::system_clock::now() ) ) );
00032   // m_JSON["meta"]["compiler"]                = nlohmann::json::meta()["compiler"];
00033   // m_JSON["meta"]["platform"]                = nlohmann::json::meta()["platform"];
00034   // m_JSON["meta"]["versions"]["json"]        = nlohmann::json::meta()["version"]["string"];
00035   // m_JSON["meta"]["versions"]["yaodaq"]      = yaodaq_version.to_string();
00036   // m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
00037 }
```

8.12.2.2 Message() [2/5]

```
yaodaq::Message::Message (
            const nlohmann::json & content,
            const MessageType & messageType = MessageType::Unknown )  [explicit]
```

Definition at line 53 of file Message.cpp.

```
00053 : Message( messageType ) { setContent( content ); }
```

8.12.2.3 Message() [3/5]

```
yaodaq::Message::Message (
            const std::string & content,
            const MessageType & messageType = MessageType::Unknown )  [explicit]
```

Definition at line 55 of file Message.cpp.

```
00055 : Message( messageType ) { setContent( content ); }
```

8.12.2.4 Message() [4/5]

```
yaodaq::Message::Message (
            const char * content,
            const MessageType & messageType = MessageType::Unknown )  [explicit]
```

Definition at line 57 of file Message.cpp.

```
00057 : Message( messageType ) { setContent( content ); }
```

8.12.2.5 Message() [5/5]

```
yaodaq::Message::Message (
            const MessageType & messageType )  [explicit], [protected]
```

Definition at line 101 of file Message.cpp.

```
00101 : Message() { m_JSON["type"] = magic_enum::enum_name( messageType ); }
```

8.12.3 Member Function Documentation

8.12.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const
```

Definition at line 59 of file Message.cpp.

```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.12.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const
```

Definition at line 61 of file Message.cpp.

```
00061 { return m_JSON; }
```

8.12.3.3 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const
```

Definition at line 67 of file Message.cpp.

```
00067 { return m_JSON["content"]; }
```

8.12.3.4 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const
```

Definition at line 89 of file Message.cpp.

```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
```

```
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
       ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00097     return id;
00098   }
00099 }
```

8.12.3.5 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const
```

Definition at line 71 of file Message.cpp.

```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
```

8.12.3.6 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const
```

Definition at line 69 of file Message.cpp.

```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.12.3.7 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const
```

Definition at line 63 of file Message.cpp.

```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.12.3.8 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const
```

Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.12.3.9 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.12.3.10 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected]
```

Definition at line 39 of file Message.cpp.

```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.12.3.11 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected]
```

Definition at line 41 of file Message.cpp.

```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

8.12.3.12 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )
```

Definition at line 80 of file Message.cpp.

```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```

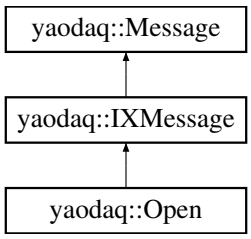
The documentation for this class was generated from the following files:

- yaodaq/Message.hpp
- yaodaq/Message.cpp

8.13 yaodaq::Open Class Reference

`#include <yaodaq/IXWebsocketMessage.hpp>`

Inheritance diagram for yaodaq::Open:



Public Member Functions

- Open (const ix::WebSocketOpenInfo &openInfo)
- Open (const ix::WebSocketOpenInfo &openInfo, std::shared_ptr< ConnectionState > &connectionState)
- std::string getURI () const
- std::map< std::string, std::string > getHeaders () const
- std::string getProtocol () const
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- void setConnectionStateInfos (std::shared_ptr< ConnectionState > &connectionState)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.13.1 Detailed Description

Definition at line 31 of file IXWebsocketMessage.hpp.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 Open() [1/2]

```
yaodaq::Open::Open (
            const ix::WebSocketOpenInfo & openInfo )  [explicit]
```

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00022                                                 : IXMessage( MessageType::Open )
00023 {
00024   nlohmann::json j = getContent();
00025   j["uri"]         = openInfo.uri;
00026   j["headers"]     = openInfo.headers;
00027   j["protocol"]    = openInfo.protocol;
00028   setContent( j );
00029 }
```

8.13.2.2 Open() [2/2]

```
yaodaq::Open::Open (
            const ix::WebSocketOpenInfo & openInfo,
            std::shared_ptr< ConnectionState > & connectionState )
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 : Open( openInfo ) { setConnectionStateInfos( connectionState ); }
```

8.13.3 Member Function Documentation

8.13.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const  [inherited]
```

Definition at line 59 of file Message.cpp.

```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.13.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const  [inherited]
```

Definition at line 61 of file Message.cpp.

```
00061 { return m_JSON; }
```

8.13.3.3 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const  [inherited]
```

Definition at line 67 of file Message.cpp.

```
00067 { return m_JSON["content"]; }
```

8.13.3.4 getHeaders()

```
std::map< std::string, std::string > yaodaq::Open::getHeaders ( ) const
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00036 {
00037   std::map<std::string, std::string> ret = get()["content"]["headers"].get<std::map<std::string,
      std::string»();
00038   return ret;
00039 }
```

8.13.3.5 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const  [inherited]
```

Definition at line 89 of file Message.cpp.

```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
      m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
      ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
      ).value() );
00097     return id;
00098   }
00099 }
```

8.13.3.6 getProtocol()

```
std::string yaodaq::Open::getProtocol ( ) const
```

Definition at line 41 of file IXWebsocketMessage.cpp.

```
00041 { return get()["content"]["protocol"].get<std::string>(); }
```

8.13.3.7 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const  [inherited]
```

Definition at line 71 of file Message.cpp.

```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
```

8.13.3.8 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const  [inherited]
```

Definition at line 69 of file Message.cpp.

```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.13.3.9 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const  [inherited]
```

Definition at line 63 of file Message.cpp.

```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.13.3.10 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const  [inherited]
```

Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.13.3.11 getURI()

```
std::string yaodaq::Open::getURI ( ) const
```

Definition at line 33 of file IXWebsocketMessage.cpp.

```
00033 { return get()["content"]["uri"].get<std::string>(); }
```

8.13.3.12 setConnectionStateInfos()

```
void yaodaq::IXMessage::setConnectionStateInfos (
            std::shared_ptr< ConnectionState > & connectionState )  [protected], [inherited]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
```

8.13.3.13 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected], [inherited]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.13.3.14 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected], [inherited]
```

Definition at line 39 of file Message.cpp.

```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.13.3.15 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected], [inherited]
```

Definition at line 41 of file Message.cpp.

```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

8.13.3.16 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )  [inherited]
```

Definition at line 80 of file Message.cpp.

```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
```

```
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```

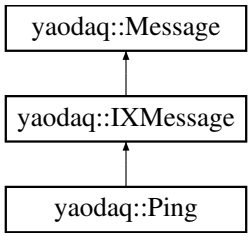
The documentation for this class was generated from the following files:

- yaodaq/IXWebsocketMessage.hpp
- yaodaq/IXWebsocketMessage.cpp

8.14 yaodaq::Ping Class Reference

`#include <yaodaq/IXWebsocketMessage.hpp>`

Inheritance diagram for yaodaq::Ping:



Public Member Functions

- Ping (const ix::WebSocketMessagePtr &ping)
- Ping (const ix::WebSocketMessagePtr &ping, std::shared_ptr< ConnectionState > &connectionState)
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- void setConnectionStateInfos (std::shared_ptr< ConnectionState > &connectionState)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.14.1 Detailed Description

Definition at line 63 of file IXWebsocketMessage.hpp.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 Ping() [1/2]

```
yaodaq::Ping::Ping (
            const ix::WebSocketMessagePtr & ping )  [explicit]
```

Definition at line 84 of file IXWebsocketMessage.cpp.

```
00084 : IXMessage( MessageType::Ping ) {}
```

8.14.2.2 Ping() [2/2]

```
yaodaq::Ping::Ping (
            const ix::WebSocketMessagePtr & ping,
            std::shared_ptr< ConnectionState > & connectionState )
```

Definition at line 86 of file IXWebsocketMessage.cpp.

```
00086 : Ping( ping ) { setConnectionStateInfos( connectionState ); }
```

8.14.3 Member Function Documentation

8.14.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const  [inherited]
```

Definition at line 59 of file Message.cpp.

```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.14.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const  [inherited]
```

Definition at line 61 of file Message.cpp.

```
00061 { return m_JSON; }
```

8.14.3.3 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const  [inherited]
```

Definition at line 67 of file Message.cpp.

```
00067 { return m_JSON["content"]; }
```

8.14.3.4 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const  [inherited]
```

Definition at line 89 of file Message.cpp.

```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
       ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00097     return id;
00098   }
00099 }
```

8.14.3.5 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const  [inherited]
```

Definition at line 71 of file Message.cpp.

```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
```

```
00078 }
```

8.14.3.6 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const  [inherited]
```

Definition at line 69 of file Message.cpp.

```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.14.3.7 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const  [inherited]
```

Definition at line 63 of file Message.cpp.

```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.14.3.8 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const  [inherited]
```

Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.14.3.9 setConnectionStateInfos()

```
void yaodaq::IXMessage::setConnectionStateInfos (
            std::shared_ptr< ConnectionState > & connectionState )  [protected], [inherited]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
```

8.14.3.10 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected], [inherited]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.14.3.11 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected], [inherited]
```

Definition at line 39 of file Message.cpp.

```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.14.3.12 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected], [inherited]
```

Definition at line 41 of file Message.cpp.

```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

8.14.3.13 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )  [inherited]
```

Definition at line 80 of file Message.cpp.

```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```

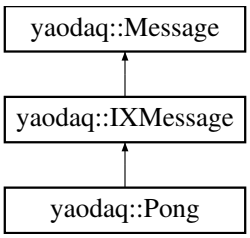
The documentation for this class was generated from the following files:

- yaodaq/IXWebsocketMessage.hpp
- yaodaq/IXWebsocketMessage.cpp

8.15 yaodaq::Pong Class Reference

`#include <yaodaq/IXWebsocketMessage.hpp>`

Inheritance diagram for yaodaq::Pong:



Public Member Functions

- Pong (const ix::WebSocketMessagePtr &pong)
- Pong (const ix::WebSocketMessagePtr &pong, std::shared_ptr< ConnectionState > &connectionState)
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

- void setConnectionStateInfos (std::shared_ptr< ConnectionState > &connectionState)
- void setContent (const nlohmann::json &content)
- void setContent (const std::string &content)
- void setContent (const char *content)

8.15.1 Detailed Description

Definition at line 70 of file IXWebsocketMessage.hpp.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 Pong() [1/2]

```
yaodaq::Pong::Pong (
            const ix::WebSocketMessagePtr & pong )  [explicit]
```

Definition at line 89 of file IXWebsocketMessage.cpp.

```
00089 : IXMessage( MessageType::Pong ) {}
```

8.15.2.2 Pong() [2/2]

```
yaodaq::Pong::Pong (
            const ix::WebSocketMessagePtr & pong,
            std::shared_ptr< ConnectionState > & connectionState )
```

Definition at line 91 of file IXWebsocketMessage.cpp.

```
00091 : Pong( pong ) { setConnectionStateInfos( connectionState ); }
```

8.15.3 Member Function Documentation

8.15.3.1 dump()

```
std::string yaodaq::Message::dump (
            const int & indent = -1,
            const char & indent_char = ' ',
            const bool & ensure_ascii = false,
            const nlohmann::detail::error_handler_t & error_handler = nlohmann::detail:↩
:error_handler_t::strict ) const  [inherited]
```

Definition at line 59 of file Message.cpp.

```
00059 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.15.3.2 get()

```
nlohmann::json yaodaq::Message::get ( ) const  [inherited]
```

Definition at line 61 of file Message.cpp.

```
00061 { return m_JSON; }
```

8.15.3.3 getContent()

```
nlohmann::json yaodaq::Message::getContent ( ) const  [inherited]
```

Definition at line 67 of file Message.cpp.

```
00067 { return m_JSON["content"]; }
```

8.15.3.4 getIdentifier()

```
Identifier yaodaq::Message::getIdentifier ( ) const  [inherited]
```

Definition at line 89 of file Message.cpp.

```
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
      m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
      ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
      ).value() );
00097     return id;
```

```
00098   }
00099 }
```

8.15.3.5 getTime()

```
std::time_t yaodaq::Message::getTime ( ) const  [inherited]
```

Definition at line 71 of file Message.cpp.

```
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
```

8.15.3.6 getTimestamp()

```
std::string yaodaq::Message::getTimestamp ( ) const  [inherited]
```

Definition at line 69 of file Message.cpp.

```
00069 { return m_JSON["timestamp"].get<std::string>(); }
```

8.15.3.7 getTypeName()

```
std::string yaodaq::Message::getTypeName ( ) const  [inherited]
```

Definition at line 63 of file Message.cpp.

```
00063 { return m_JSON["type"].get<std::string>(); }
```

8.15.3.8 getTypeValue()

```
MessageType yaodaq::Message::getTypeValue ( ) const  [inherited]
```

Definition at line 65 of file Message.cpp.

```
00065 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.15.3.9 setConnectionStateInfos()

```
void yaodaq::IXMessage::setConnectionStateInfos (
            std::shared_ptr< ConnectionState > & connectionState )  [protected], [inherited]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
```

8.15.3.10 setContent() [1/3]

```
void yaodaq::Message::setContent (
            const char * content )  [protected], [inherited]
```

Definition at line 47 of file Message.cpp.

```
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
```

8.15.3.11 setContent() [2/3]

```
void yaodaq::Message::setContent (
            const nlohmann::json & content )  [protected], [inherited]
```

Definition at line 39 of file Message.cpp.
```
00039 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.15.3.12 setContent() [3/3]

```
void yaodaq::Message::setContent (
            const std::string & content )  [protected], [inherited]
```
Definition at line 41 of file Message.cpp.
```
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
```

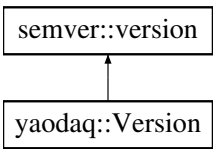
8.15.3.13 setFrom()

```
void yaodaq::Message::setFrom (
            const Identifier & identifier )  [inherited]
```
Definition at line 80 of file Message.cpp.
```
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
```
The documentation for this class was generated from the following files:

- yaodaq/IXWebsocketMessage.hpp
- yaodaq/IXWebsocketMessage.cpp

8.16 yaodaq::Version Class Reference

`#include <yaodaq/Version.hpp>`
Inheritance diagram for yaodaq::Version:



Public Member Functions

- constexpr Version (const std::uint8_t &mj, const std::uint8_t &mn, const std::uint8_t &pt, const semver↩ ::prerelease &prt=semver::prerelease::none, const std::uint8_t &prn=0) noexcept
- constexpr Version (const std::string_view &str)
- constexpr Version ()=default
- std::uint8_t getMajor ()
- std::uint8_t getMinor ()
- std::uint8_t getPatch ()
- std::string getPreRelease ()
- std::uint8_t getPreReleaseNumber ()

8.16.1 Detailed Description

Definition at line 15 of file Version.hpp.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 Version() [1/3]

```
constexpr yaodaq::Version::Version (
            const std::uint8_t & mj,
            const std::uint8_t & mn,
            const std::uint8_t & pt,
            const semver::prerelease & prt = semver::prerelease::none,
            const std::uint8_t & prn = 0 )  [inline], [constexpr], [noexcept]
```

Definition at line 18 of file Version.hpp.

```
00018 : semver::version( mj, mn, pt, prt, prn ) {}
```

8.16.2.2 Version() [2/3]

```
constexpr yaodaq::Version::Version (
            const std::string_view & str )  [inline], [explicit], [constexpr]
```

Definition at line 19 of file Version.hpp.

```
00019 : semver::version( str ) {}
```

8.16.2.3 Version() [3/3]

```
constexpr yaodaq::Version::Version ( )  [constexpr], [default]
```

8.16.3 Member Function Documentation

8.16.3.1 getMajor()

```
std::uint8_t yaodaq::Version::getMajor ( )
```

Definition at line 12 of file Version.cpp.

```
00012 { return major; }
```

8.16.3.2 getMinor()

```
std::uint8_t yaodaq::Version::getMinor ( )
```

Definition at line 14 of file Version.cpp.

```
00014 { return minor; }
```

8.16.3.3 getPatch()

```
std::uint8_t yaodaq::Version::getPatch ( )
```

Definition at line 16 of file Version.cpp.

```
00016 { return patch; }
```

8.16.3.4 getPreRelease()

```
std::string yaodaq::Version::getPreRelease ( )
```

Definition at line 18 of file Version.cpp.

```
00018 { return std::string( magic_enum::enum_name( prerelease_type ) ); }
```

8.16.3.5 getPreReleaseNumber()

```
std::uint8_t yaodaq::Version::getPreReleaseNumber ( )
```

Definition at line 20 of file Version.cpp.

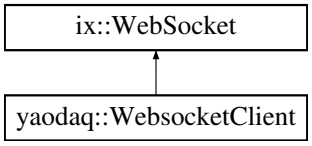
```
00020 { return prerelease_number; }
```

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

- yaodaq/Version.hpp
- yaodaq/Version.cpp

8.17 yaodaq::WebsocketClient Class Reference

`#include <yaodaq/WebsocketClient.hpp>`
Inheritance diagram for yaodaq::WebsocketClient:



Public Member Functions

- WebsocketClient (const std::string &name, const std::string &type="YAODAQWebsocketClient")
- virtual ~WebsocketClient ()
- void start ()
- void stop ()
- void loop ()
- std::shared_ptr< spdlog::logger > logger ()
- virtual void onMessage (Message &message)
- virtual void onOpen (Open &open)
- virtual void onClose (Close &close)
- virtual void onError (Error &error)
- virtual void onPing (Ping &ping)
- virtual void onPong (Pong &pong)
- virtual void onFragment (Fragment &fragment)

8.17.1 Detailed Description

Definition at line 29 of file WebsocketClient.hpp.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 WebsocketClient()

```
yaodaq::WebsocketClient::WebsocketClient (
            const std::string & name,
            const std::string & type = "YAODAQWebsocketClient" )  [explicit]
```

Definition at line 20 of file WebsocketClient.cpp.

```
00020                                                                   : m_Identifier( type,
      name )
00021 {
00022   ix::initNetSystem();
00023
00024   m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
00025   m_Logger.setName( m_Identifier.get() );
00026   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00027
00028   ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
00029   setExtraHeaders( header );
00030
00031   setOnMessageCallback(
00032     [this]( const ix::WebSocketMessagePtr& msg )
00033     {
00034       if( msg->type == ix::WebSocketMessageType::Message ) { logger()->error( "{}", msg->str ); }
00035       else if( msg->type == ix::WebSocketMessageType::Error )
00036       {
```

```
00037         std::cout « "Connection error: " « msg->errorInfo.reason « std::endl;
00038       }
00039       else if( msg->type == ix::WebSocketMessageType::Close )
00040       {
00041         disableAutomaticReconnection();
00042         if( msg->closeInfo.code == magic_enum::enum_integer(
      StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ) )
00043         {
00044           logger()->critical( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold,
      msg->closeInfo.reason ) );
00045           close();
00046           // throw Exception( StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED,
      msg->closeInfo.reason );
00047         }
00048       }
00049     }
00050
00051   );
00052 }
```

8.17.2.2 ~WebsocketClient()

`yaodaq::WebsocketClient::~WebsocketClient ( )  [virtual]`

Definition at line 68 of file WebsocketClient.cpp.

```
00069 {
00070   stop();
00071   ix::uninitNetSystem();
00072 }
```

8.17.3 Member Function Documentation

8.17.3.1 logger()

`std::shared_ptr< spdlog::logger > yaodaq::WebsocketClient::logger ( )  [inline]`

Definition at line 37 of file WebsocketClient.hpp.

```
00037 { return m_Logger.logger(); }
```

8.17.3.2 loop()

`void yaodaq::WebsocketClient::loop ( )`

Definition at line 93 of file WebsocketClient.cpp.

```
00094 {
00095   WebsocketClient::start();
00096   m_Looper.supressInstance();
00097   onRaisingSignal();
00098 }
```

8.17.3.3 onClose()

`void yaodaq::WebsocketClient::onClose (`
`Close & close )  [virtual]`

Definition at line 58 of file WebsocketClient.cpp.

```
00058 {}
```

8.17.3.4 onError()

`void yaodaq::WebsocketClient::onError (`
`Error & error )  [virtual]`

Definition at line 60 of file WebsocketClient.cpp.

```
00060 {}
```

8.17.3.5 onFragment()

```
void yaodaq::WebsocketClient::onFragment (
            Fragment & fragment )  [virtual]
```

Definition at line 66 of file WebsocketClient.cpp.

```
00066 {}
```

8.17.3.6 onMessage()

```
void yaodaq::WebsocketClient::onMessage (
            Message & message )  [virtual]
```

Definition at line 54 of file WebsocketClient.cpp.

```
00054 {}
```

8.17.3.7 onOpen()

```
void yaodaq::WebsocketClient::onOpen (
            Open & open )  [virtual]
```

Definition at line 56 of file WebsocketClient.cpp.

```
00056 {}
```

8.17.3.8 onPing()

```
void yaodaq::WebsocketClient::onPing (
            Ping & ping )  [virtual]
```

Definition at line 62 of file WebsocketClient.cpp.

```
00062 {}
```

8.17.3.9 onPong()

```
void yaodaq::WebsocketClient::onPong (
            Pong & pong )  [virtual]
```

Definition at line 64 of file WebsocketClient.cpp.

```
00064 {}
```

8.17.3.10 start()

```
void yaodaq::WebsocketClient::start ( )
```

Definition at line 74 of file WebsocketClient.cpp.

```
00075 {
00076   if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00077   {
00078     logger()->trace( "Client started. Connected to {}", getUrl() );
00079     ix::WebSocket::start();
00080   }
00081 }
```

8.17.3.11 stop()

```
void yaodaq::WebsocketClient::stop ( )
```

Definition at line 83 of file WebsocketClient.cpp.

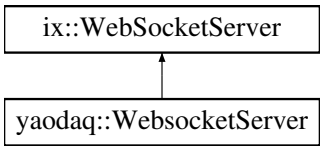
```
00084 {
00085   if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00086   {
00087     logger()->trace( "Client stopped" );
00088     ix::WebSocket::stop();
00089     while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
       std::chrono::microseconds( 1 ) ); }
00090   }
00091 }
```

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

- yaodaq/WebsocketClient.hpp
- yaodaq/WebsocketClient.cpp

8.18 yaodaq::WebsocketServer Class Reference

`#include <yaodaq/WebsocketServer.hpp>`
Inheritance diagram for yaodaq::WebsocketServer:



Public Member Functions

- WebsocketServer (const std::string &name, const int &port=ix::SocketServer::kDefaultPort, const std::string &host=ix::SocketServer::kDefaultHost, const int &backlog=ix::SocketServer::kDefaultTcpBacklog, const std::size_t &maxConnections=ix::SocketServer::kDefaultMaxConnections, const int &handshakeTimeout↩Secs=ix::WebSocketServer::kDefaultHandShakeTimeoutSecs, const int &addressFamily=ix::SocketServer↩::kDefaultAddressFamily, const std::string &type="YAODAQWebsocketServer")
- virtual ~WebsocketServer ()
- void loop ()
- void start ()
- void stop (bool useless=true)
- void listen ()
- virtual void onMessage (Message &message)
- virtual void onOpen (Open &open)
- virtual void onClose (Close &close)
- virtual void onError (Error &error)
- virtual void onPing (Ping &ping)
- virtual void onPong (Pong &pong)
- virtual void onFragment (Fragment &fragment)
- void setVerbosity (const yaodaq::LoggerHandler::Verbosity &verbosity)
- std::shared_ptr< spdlog::logger > logger ()
- void sendToLoggers (Message &message)
- void sendToLoggers (const Message &message)
- void sendToLoggers (Message &message, ix::WebSocket &webSocket)
- void sendToLoggers (const Message &message, ix::WebSocket &webSocket)

8.18.1 Detailed Description

Definition at line 31 of file WebsocketServer.hpp.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 WebsocketServer()

```
yaodaq::WebsocketServer::WebsocketServer (
            const std::string & name,
            const int & port = ix::SocketServer::kDefaultPort,
            const std::string & host = ix::SocketServer::kDefaultHost,
            const int & backlog = ix::SocketServer::kDefaultTcpBacklog,
```

```
            const std::size_t & maxConnections = ix::SocketServer::kDefaultMaxConnections,
            const int & handshakeTimeoutSecs = ix::WebSocketServer::kDefaultHandShakeTimeoutSecs,
            const int & addressFamily = ix::SocketServer::kDefaultAddressFamily,
            const std::string & type = "YAODAQWebsocketServer" )  [explicit]
```

Definition at line 27 of file WebsocketServer.cpp.

```
00027
                                :
00028   ix::WebSocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
       m_Identifier( type, name )
00029 {
00030   ix::initNetSystem();
00031
00032   m_Identifier.generateKey( Domain::Application, Class::Server, Family::WebSocketServer );
00033   m_Logger.setName( m_Identifier.get() );
00034   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00035
00036   setConnectionStateFactory( []() { return std::make_shared<ConnectionState>(); } );
00037
00038   setOnClientMessageCallback(
00039     [this]( std::shared_ptr<ix::ConnectionState> connectionState, ix::WebSocket& webSocket, const
       ix::WebSocketMessagePtr& msg )
00040     {
00041       // The ConnectionState object contains information about the connection
00042       std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>(
       connectionState );
00043       if( msg->type == ix::WebSocketMessageType::Message ) {}
00044       else if( msg->type == ix::WebSocketMessageType::Open )
00045       {
00046         // Check if a client with the same name is already connected;
00047         connection->computeId( getHost() + ":" + std::to_string( getPort() ), Identifier::parse(
       msg->openInfo.headers["id"] ) );
00048         if( connection->isTerminated() )
00049         {
00050           logger()->error( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold, "One client with
       the name \"{}\" is already connected !", Identifier::parse( msg->openInfo.headers["id"] ).getName() )
       );
00051           webSocket.stop( magic_enum::enum_integer(
       StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ),
00052                           fmt::format( "One client with the name \"{}\" is already connected to
       ws{}://{}:{} !", Identifier::parse( msg->openInfo.headers["id"] ).getName(), "", getHost(), getPort()
       ) );
00053           std::this_thread::sleep_for( std::chrono::milliseconds( 10 ) );
00054           return;
00055         }
00056         addClient( Identifier::parse( msg->openInfo.headers["id"] ), webSocket );
00057         Open open( msg->openInfo, connection );
00058         sendToLoggers( open, webSocket );
00059         onOpen( open );
00060       }
00061       else if( msg->type == ix::WebSocketMessageType::Close )
00062       {
00063         Close close( msg->closeInfo, connection );
00064         sendToLoggers( close, webSocket );
00065         onClose( close );
00066         removeClient( webSocket );
00067       }
00068       else if( msg->type == ix::WebSocketMessageType::Error )
00069       {
00070         Error error( msg->errorInfo, connection );
00071         sendToLoggers( error, webSocket );
00072         onError( error );
00073       }
00074       else if( msg->type == ix::WebSocketMessageType::Ping )
00075       {
00076         Ping ping( msg, connection );
00077         sendToLoggers( ping, webSocket );
00078         onPing( ping );
00079       }
00080       else if( msg->type == ix::WebSocketMessageType::Pong )
00081       {
00082         Pong pong( msg, connection );
00083         sendToLoggers( pong, webSocket );
00084         onPong( pong );
00085       }
00086       else if( msg->type == ix::WebSocketMessageType::Fragment )
00087       {
00088         Fragment fragment( msg, connection );
00089         sendToLoggers( fragment, webSocket );
00090         onFragment( fragment );
00091       }
00092     } );
00093 }
```

8.18.2.2 ~WebsocketServer()

```
yaodaq::WebsocketServer::~WebsocketServer ( )  [virtual]
```

Definition at line 198 of file WebsocketServer.cpp.

```
00199 {
00200   stop();
00201   ix::uninitNetSystem();
00202 }
```

8.18.3 Member Function Documentation

8.18.3.1 listen()

```
void yaodaq::WebsocketServer::listen ( )
```

Definition at line 160 of file WebsocketServer.cpp.

```
00161 {
00162   if( !m_isListening )
00163   {
00164     std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
00165     if( ret.first )
00166     {
00167       m_isListening = ret.first;
00168       logger()->info( "Server listening on {0}:{1}", getHost(), getPort() );
00169     }
00170     else
00171       throw Exception( StatusCode::LISTEN_ERROR, ret.second );
00172   }
00173 }
```

8.18.3.2 logger()

```
std::shared_ptr< spdlog::logger > yaodaq::WebsocketServer::logger ( )  [inline]
```

Definition at line 53 of file WebsocketServer.hpp.

```
00053 { return m_Logger.logger(); }
```

8.18.3.3 loop()

```
void yaodaq::WebsocketServer::loop ( )
```

Definition at line 204 of file WebsocketServer.cpp.

```
00205 {
00206   listen();
00207   start();
00208   m_Looper.supressInstance();
00209   onRaisingSignal();
00210 }
```

8.18.3.4 onClose()

```
void yaodaq::WebsocketServer::onClose (
            Close & close )  [virtual]
```

Definition at line 150 of file WebsocketServer.cpp.

```
00150 {}
```

8.18.3.5 onError()

```
void yaodaq::WebsocketServer::onError (
            Error & error )  [virtual]
```

Definition at line 152 of file WebsocketServer.cpp.

```
00152 {}
```

8.18.3.6 onFragment()

```
void yaodaq::WebsocketServer::onFragment (
            Fragment & fragment )  [virtual]
```

Definition at line 158 of file WebsocketServer.cpp.

```
00158 {}
```

8.18.3.7 onMessage()

```
void yaodaq::WebsocketServer::onMessage (
            Message & message )  [virtual]
```

Definition at line 146 of file WebsocketServer.cpp.

```
00146 {}
```

8.18.3.8 onOpen()

```
void yaodaq::WebsocketServer::onOpen (
            Open & open )  [virtual]
```

Definition at line 148 of file WebsocketServer.cpp.

```
00148 {}
```

8.18.3.9 onPing()

```
void yaodaq::WebsocketServer::onPing (
            Ping & ping )  [virtual]
```

Definition at line 154 of file WebsocketServer.cpp.

```
00154 {}
```

8.18.3.10 onPong()

```
void yaodaq::WebsocketServer::onPong (
            Pong & pong )  [virtual]
```

Definition at line 156 of file WebsocketServer.cpp.

```
00156 {}
```

8.18.3.11 sendToLoggers() [1/4]

```
void yaodaq::WebsocketServer::sendToLoggers (
            const Message & message )
```

Definition at line 138 of file WebsocketServer.cpp.

```
00139 {
00140   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00141   {
00142     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
      it->second.send( message.dump() );
00143   }
00144 }
```

8.18.3.12 sendToLoggers() [2/4]

```
void yaodaq::WebsocketServer::sendToLoggers (
            const Message & message,
            ix::WebSocket & webSocket )
```

Definition at line 130 of file WebsocketServer.cpp.

```
00131 {
00132   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00133   {
```

```
00134     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
       != &it->second ) it->second.send( message.dump() );
00135   }
00136 }
```

8.18.3.13 sendToLoggers() [3/4]

```
void yaodaq::WebsocketServer::sendToLoggers (
              Message & message )
```

Definition at line 122 of file WebsocketServer.cpp.

```
00123 {
00124   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
       ++it )
00125   {
00126     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
       it->second.send( message.dump() );
00127   }
00128 }
```

8.18.3.14 sendToLoggers() [4/4]

```
void yaodaq::WebsocketServer::sendToLoggers (
              Message & message,
              ix::WebSocket & webSocket )
```

Definition at line 114 of file WebsocketServer.cpp.

```
00115 {
00116   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
       ++it )
00117   {
00118     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
       != &it->second ) it->second.send( message.dump() );
00119   }
00120 }
```

8.18.3.15 setVerbosity()

```
void yaodaq::WebsocketServer::setVerbosity (
              const yaodaq::LoggerHandler::Verbosity & verbosity )
```

Definition at line 196 of file WebsocketServer.cpp.

```
00196 { m_Logger.setVerbosity( verbosity ); }
```

8.18.3.16 start()

```
void yaodaq::WebsocketServer::start ( )
```

Definition at line 175 of file WebsocketServer.cpp.

```
00176 {
00177   if( !m_isStarted )
00178   {
00179     m_isStarted = true;
00180     logger()->trace( "Server started" );
00181     ix::WebSocketServer::start();
00182   }
00183 }
```

8.18.3.17 stop()

```
void yaodaq::WebsocketServer::stop (
              bool useless = true )
```

Definition at line 185 of file WebsocketServer.cpp.

```
00186 {
00187   if( !m_isStopped )
00188   {
00189     m_isStopped = true;
00190     useless     = !useless;
00191     logger()->trace( "Server stopped" );
00192     ix::WebSocketServer::stop();
```

```
00193   }
00194 }
```

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

- yaodaq/WebsocketServer.hpp
- yaodaq/WebsocketServer.cpp

Chapter 9

File Documentation

9.1 docs/License.md File Reference

9.2 docs/Third-party licenses.md File Reference

9.3 yaodaq/Classification.hpp File Reference

```
#include <cstdint>
```

Namespaces

- namespace yaodaq

Enumerations

- enum class yaodaq::Domain : std::uint_least8_t { yaodaq::Unknown = 0 , yaodaq::Application = 1 , yaodaq::Web = 2 }
- enum class yaodaq::Class : std::uint_least8_t { yaodaq::Unknown = 0 , yaodaq::Server , yaodaq::Client , yaodaq::Module , yaodaq::Board }
- enum class yaodaq::Family : std::uint_least16_t { yaodaq::Unknown = 0 , yaodaq::WebSocketServer , yaodaq::WebSocketClient , yaodaq::Logger , yaodaq::Controller , yaodaq::Configurator , yaodaq::SlowController , yaodaq::Viewer , yaodaq::Analyser , yaodaq::FileWriter }

9.4 Classification.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_CLASSIFICATION_HPP
00002 #define YAODAQ_CLASSIFICATION_HPP
00003
00008 #include <cstdint>
00009
00010 namespace yaodaq
00011 {
00012
00013 /* The domain specify if we are on browser or standalone program */
00014 enum class Domain : std::uint_least8_t
00015 {
00016   Unknown     = 0,
00017   Application = 1,
00018   Web         = 2,
00019 };
00020
00021 /* The class define if we are a server, module, or board */
00022 enum class Class : std::uint_least8_t
00023 {
00024   Unknown = 0,
```

```
00025   Server,
00026   Client,
00027   // Module is a client with start stop etc...
00028   Module,
00029   // Board is a module with a connector
00030   Board,
00031 };
00032
00033 /* the family */
00034 enum class Family : std::uint_least16_t
00035 {
00036   Unknown = 0,
00037   WebSocketServer,
00038   WebSocketClient,
00039   Logger,
00040   Controller,
00041   Configurator,
00042   SlowController,
00043   Viewer,
00044   Analyser,
00045   FileWriter,
00046 };
00047
00048 }  // namespace yaodaq
00049
00050 #endif  // YAODAQ_CLASSIFICATION_HPP
```

9.5 yaodaq/ConnectionState.hpp File Reference

```
#include <algorithm>
#include <iostream>
#include <ixwebsocket/IXConnectionState.h>
#include <list>
#include <mutex>
#include <string>
#include <utility>
```

Data Structures

- class yaodaq::ConnectionState

Namespaces

- namespace yaodaq

9.6 ConnectionState.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_CONNECTIONSTATE
00002 #define YAODAQ_CONNECTIONSTATE
00003
00008 #include <algorithm>
00009 #include <iostream>
00010 #include <ixwebsocket/IXConnectionState.h>
00011 #include <list>
00012 #include <mutex>
00013 #include <string>
00014 #include <utility>
00015
00016 namespace yaodaq
00017 {
00018
00019 class Identifier;
00020
00021 class ConnectionState : public ix::ConnectionState
00022 {
00023 public:
00024   virtual void computeId( const std::string& host, const Identifier& id ) final;
00025   ConnectionState();
00026   virtual ~ConnectionState();
00027
00028 private:
```

```
00029   static std::list<std::pair<std::string, std::string>> m_Ids;
00030   std::pair<std::string, std::string>                   m_Pair;
00031   std::mutex                                            m_Mutex;
00032 };
00033
00034 }  // namespace yaodaq
00035
00036 #endif
```

9.7 yaodaq/Exception.hpp File Reference

```
#include <cstdint>
#include <exception>
#include <fmt/color.h>
#include <source_location/source_location.hpp>
#include <string>
```

Data Structures

- class yaodaq::Exception

Namespaces

- namespace yaodaq

9.8 Exception.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_EXCEPTION
00002 #define YAODAQ_EXCEPTION
00003
00008 #include <cstdint>
00009 #include <exception>
00010 #include <fmt/color.h>
00011 #include <source_location/source_location.hpp>
00012 #include <string>
00013
00014 namespace yaodaq
00015 {
00016
00017 enum class StatusCode : std::int_least32_t;
00018
00019 class Exception : public std::exception, public source_location
00020 {
00021 public:
00022   Exception() = delete;
00023
00024   static void setFormat( const std::string& format ) { m_Format = format; }
00025
00026   static void setStyle( const fmt::text_style& style = {} ) { m_Style = style; }
00027
00028   Exception( const StatusCode& statusCode, const std::string& description, const source_location&
      location = source_location::current() );
00029   ~Exception() noexcept override = default;
00030   [[nodiscard]] const char*        what() const noexcept final;
00031   [[nodiscard]] const char*        description() const noexcept;
00032   [[nodiscard]] std::int_least32_t code() const noexcept;
00033
00034 private:
00035   static fmt::text_style   m_Style;
00036   static std::string       m_Format;
00037   const std::int_least32_t m_Code{ 0 };
00038   std::string              m_Description;
00039   std::string              m_Message;
00040   void                     constructMessage();
00041 };
00042
00043 }  // namespace yaodaq
00044
00045 #endif
```

9.9 yaodaq/Identifier.hpp File Reference

```
#include "yaodaq/Key.hpp"
#include <cstdint>
#include <string>
```

Data Structures

- class yaodaq::Identifier

Namespaces

- namespace yaodaq

9.10 Identifier.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_IDENTIFIER_HPP
00002 #define YAODAQ_IDENTIFIER_HPP
00003
00008 #include "yaodaq/Key.hpp"
00009
00010 #include <cstdint>
00011 #include <string>
00012
00013 namespace yaodaq
00014 {
00015
00016 class Identifier
00017 {
00018 public:
00019   Identifier() = default;
00020   Identifier( const std::string& type, const std::string& name );
00021   void                    generateKey( const Domain& domain = Domain::Unknown, const Class& c_lass =
       Class::Unknown, const Family& family = Family::Unknown );
00022   [[nodiscard]] std::string getDomain() const;
00023   [[nodiscard]] std::string getClass() const;
00024   [[nodiscard]] std::string getFamily() const;
00025   [[nodiscard]] std::string getType() const;
00026   [[nodiscard]] std::string getName() const;
00027   [[nodiscard]] Key         getKey() const;
00028   [[nodiscard]] std::string get() const;
00029   bool                    empty() const;
00030   static Identifier       parse( const std::string& );
00031   bool                    operator<( const Identifier& ) const;
00032
00033 private:
00034   std::string m_Type{ "Unknown" };
00035   std::string m_Name{ "Unknown" };
00036   Key         m_Key;
00037 };
00038
00039 }  // namespace yaodaq
00040
00041 #endif  // YAODAQ_IDENTIFIER_HPP
```

9.11 yaodaq/Interrupt.hpp File Reference

```
#include "yaodaq/Signal.hpp"
#include <atomic>
#include <csignal>
#include <mutex>
```

Data Structures

- class yaodaq::Interrupt

Namespaces

- namespace yaodaq

9.12 Interrupt.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_HANDLER_HPP
00002 #define YAODAQ_HANDLER_HPP
00003
00008 #include "yaodaq/Signal.hpp"
00009
00010 #include <atomic>
00011 #include <csignal>
00012 #include <mutex>
00013
00014 namespace yaodaq
00015 {
00016
00017 enum class Signal;
00018
00019 class Interrupt
00020 {
00021 public:
00022   Interrupt();
00023   void   init();
00024   void   restore();
00025   Signal getSignal();
00026   ~Interrupt();
00027
00028 private:
00029   volatile static std::atomic<Signal> m_Signal;
00030   void                                setSignal( const Signal& signal );
00031   std::mutex                          m_mutex;
00032 };
00033
00034 }  // namespace yaodaq
00035
00036 #endif  // YAODAQ_HANDLER_HPP
```

9.13 yaodaq/IXWebsocketMessage.hpp File Reference

```
#include "yaodaq/ConnectionState.hpp"
#include "yaodaq/Message.hpp"
#include <ixwebsocket/IXWebSocketCloseInfo.h>
#include <ixwebsocket/IXWebSocketErrorInfo.h>
#include <ixwebsocket/IXWebSocketMessage.h>
#include <ixwebsocket/IXWebSocketOpenInfo.h>
#include <map>
#include <memory>
#include <string>
```

Data Structures

- class yaodaq::IXMessage
- class yaodaq::Open
- class yaodaq::Close
- class yaodaq::Error
- class yaodaq::Ping
- class yaodaq::Pong
- class yaodaq::Fragment

Namespaces

- namespace yaodaq

9.14 IXWebsocketMessage.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_IXWEBSOCKETMESSAGE
00002 #define YAODAQ_IXWEBSOCKETMESSAGE
00003
00008 #include "yaodaq/ConnectionState.hpp"
00009 #include "yaodaq/Message.hpp"
00010
00011 #include <ixwebsocket/IXWebSocketCloseInfo.h>
00012 #include <ixwebsocket/IXWebSocketErrorInfo.h>
00013 #include <ixwebsocket/IXWebSocketMessage.h>
00014 #include <ixwebsocket/IXWebSocketOpenInfo.h>
00015 #include <map>
00016 #include <memory>
00017 #include <string>
00018
00019 namespace yaodaq
00020 {
00021
00022 class IXMessage : public Message
00023 {
00024 public:
00025   explicit IXMessage( const MessageType& messageType );
00026
00027 protected:
00028   void setConnectionStateInfos( std::shared_ptr<ConnectionState>& connectionState );
00029 };
00030
00031 class Open : public IXMessage
00032 {
00033 public:
00034   explicit Open( const ix::WebSocketOpenInfo& openInfo );
00035   Open( const ix::WebSocketOpenInfo& openInfo, std::shared_ptr<ConnectionState>& connectionState );
00036   std::string                        getURI() const;
00037   std::map<std::string, std::string> getHeaders() const;
00038   std::string                        getProtocol() const;
00039 };
00040
00041 class Close : public IXMessage
00042 {
00043 public:
00044   explicit Close( const ix::WebSocketCloseInfo& closeInfo );
00045   Close( const ix::WebSocketCloseInfo& closeInfo, std::shared_ptr<ConnectionState>& connectionState );
00046   std::uint16_t getCode() const;
00047   std::string   getReason() const;
00048   bool          getRemote() const;
00049 };
00050
00051 class Error : public IXMessage
00052 {
00053 public:
00054   explicit Error( const ix::WebSocketErrorInfo& errorInfo );
00055   Error( const ix::WebSocketErrorInfo& errorInfo, std::shared_ptr<ConnectionState>& connectionState );
00056   std::uint16_t getRetries() const;
00057   double        getWaitTime() const;
00058   int           getHttpStatus() const;
00059   std::string   getReason() const;
00060   bool          getDecompressionError() const;
00061 };
00062
00063 class Ping : public IXMessage
00064 {
00065 public:
00066   explicit Ping( const ix::WebSocketMessagePtr& ping );
00067   Ping( const ix::WebSocketMessagePtr& ping, std::shared_ptr<ConnectionState>& connectionState );
00068 };
00069
00070 class Pong : public IXMessage
00071 {
00072 public:
00073   explicit Pong( const ix::WebSocketMessagePtr& pong );
00074   Pong( const ix::WebSocketMessagePtr& pong, std::shared_ptr<ConnectionState>& connectionState );
00075 };
00076
00077 class Fragment : public IXMessage
00078 {
00079 public:
00080   explicit Fragment( const ix::WebSocketMessagePtr& fragment );
00081   Fragment( const ix::WebSocketMessagePtr& fragment, std::shared_ptr<ConnectionState>& connectionState
      );
00082 };
00083
00084 }  // namespace yaodaq
00085 #endif
```

9.15 yaodaq/Key.hpp File Reference

```
#include "yaodaq/Classification.hpp"
#include <cstdint>
```

Data Structures

- class yaodaq::Key

Namespaces

- namespace yaodaq

9.16 Key.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_KEY_HPP
00002 #define YAODAQ_KEY_HPP
00003
00008 #include "yaodaq/Classification.hpp"
00009
00010 #include <cstdint>
00011
00012 namespace yaodaq
00013 {
00014
00015 class Key
00016 {
00017 private:
00018   std::int_least32_t m_Key{ 0 };
00019
00020 public:
00021   Key() = default;
00022   explicit Key( const Domain& domain, const Class& c_lass, const Family& family );
00023   [[nodiscard]] std::int_least8_t  getDomain() const;
00024   [[nodiscard]] std::int_least8_t  getClass() const;
00025   [[nodiscard]] std::int_least16_t getFamily() const;
00026   [[nodiscard]] std::int_least32_t getKey() const;
00027 };
00028
00029 }  // namespace yaodaq
00030
00031 #endif  // YAODAQ_KEY_HPP
```

9.17 yaodaq/LoggerHandler.hpp File Reference

```
#include <memory>
#include <spdlog/fwd.h>
#include <string>
#include <vector>
```

Data Structures

- class yaodaq::LoggerHandler

Namespaces

- namespace spdlog
- namespace yaodaq

Typedefs

- using spdlog::sink_ptr = std::shared_ptr< spdlog::sinks::sink >

9.18 LoggerHandler.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_LOGGERHANDLER
00002 #define YAODAQ_LOGGERHANDLER
00003
00008 #include <memory>
00009 #include <spdlog/fwd.h>
00010 #include <string>
00011 #include <vector>
00012
00013 namespace spdlog
00014 {
00015 using sink_ptr = std::shared_ptr<spdlog::sinks::sink>;
00016 }
00017
00018 namespace yaodaq
00019 {
00020
00021 class LoggerHandler
00022 {
00023 public:
00024   enum class Verbosity
00025   {
00026     Off,
00027     Trace,
00028     Debug,
00029     Info,
00030     Warn,
00031     Error,
00032     Critical
00033   };
00034   LoggerHandler();
00035   ~LoggerHandler();
00036   void                              setVerbosity( const Verbosity& verbosity );
00037   void                              setName( const std::string& );
00038   std::shared_ptr<spdlog::logger> logger();
00039   void                              addSink( const spdlog::sink_ptr& );
00040   void                              clearSinks();
00041
00042 private:
00043   std::shared_ptr<spdlog::logger> m_Logger{ nullptr };
00044   std::vector<spdlog::sink_ptr>   m_Sinks;
00045   std::string                     m_Name{ "Unknown" };
00046   Verbosity                       m_Verbosity{ Verbosity::Trace };
00047   void                            init();
00048 };
00049
00050 }  // namespace yaodaq
00051
00052 #endif
```

9.19 yaodaq/Looper.hpp File Reference

```
#include "yaodaq/Interrupt.hpp"
```

Data Structures

- class yaodaq::Looper

Namespaces

- namespace yaodaq

9.20 Looper.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_LOOPER
00002 #define YAODAQ_LOOPER
00003
00008 #include "yaodaq/Interrupt.hpp"
00009
00010 namespace yaodaq
00011 {
```

```
00012
00013 enum class Signal;
00014
00015 class Looper
00016 {
00017 public:
00018   Looper();
00019   Signal     loop();
00020   Signal     getSignal();
00021   static int getInstance();
00022   void       supressInstance();
00023   ~Looper();
00024
00025 private:
00026   static int       m_instance;
00027   bool             m_hasBeenAdded{ false };
00028   bool             m_hasBeenSupressed{ false };
00029   static Interrupt m_Interrupt;
00030 };
00031
00032 }  // namespace yaodaq
00033
00034 #endif  // YAODAQ_LOOPER
```

9.21 yaodaq/Message.hpp File Reference

```
#include "nlohmann/json.hpp"
#include "yaodaq/MessageType.hpp"
#include <string>
```

Data Structures

- class yaodaq::Message

Namespaces

- namespace yaodaq

9.22 Message.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_MESSAGE
00002 #define YAODAQ_MESSAGE
00003
00008 #include "nlohmann/json.hpp"
00009 #include "yaodaq/MessageType.hpp"
00010
00011 #include <string>
00012
00013 namespace yaodaq
00014 {
00015
00016 class Identifier;
00017
00018 class Message
00019 {
00020 public:
00021   Message();
00022   explicit Message( const nlohmann::json& content, const MessageType& messageType =
      MessageType::Unknown );
00023   explicit Message( const std::string& content, const MessageType& messageType = MessageType::Unknown
      );
00024   explicit Message( const char* content, const MessageType& messageType = MessageType::Unknown );
00025   std::string    dump( const int& indent = -1, const char& indent_char = ' ', const bool& ensure_ascii
      = false, const nlohmann::detail::error_handler_t& error_handler =
      nlohmann::detail::error_handler_t::strict ) const;
00026   nlohmann::json get() const;
00027   nlohmann::json getContent() const;
00028   std::string    getTypeName() const;
00029   MessageType    getTypeValue() const;
00030   std::string    getTimestamp() const;
00031   std::time_t    getTime() const;
00032   Identifier     getIdentifier() const;
00033   void           setFrom( const Identifier& );
```

```
00034
00035 protected:
00036   explicit Message( const MessageType& messageType );
00037   void setContent( const nlohmann::json& content );
00038   void setContent( const std::string& content );
00039   void setContent( const char* content );
00040
00041 private:
00042   nlohmann::json m_JSON;
00043 };
00044
00045 }  // namespace yaodaq
00046
00047 #endif  // YAODAQ_MESSAGE
```

9.23 yaodaq/MessageType.hpp File Reference

```
#include "yaodaq/Interrupt.hpp"
#include <cstdint>
#include <iosfwd>
```

Namespaces

- namespace yaodaq

Enumerations

- enum class yaodaq::MessageType : std::int_least16_t { yaodaq::Open = -1 , yaodaq::Close = -2 , yaodaq::Error = -3 , yaodaq::Ping = -4 , yaodaq::Pong = -5 , yaodaq::Fragment = -6 , yaodaq::Unknown = 0 }

Functions

- std::ostream & yaodaq::operator<< (std::ostream &os, const MessageType &messageTypes)

9.24 MessageType.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_MESSAGETYPE
00002 #define YAODAQ_MESSAGETYPE
00003
00007 #include "yaodaq/Interrupt.hpp"
00008
00009 #include <cstdint>
00010 #include <iosfwd>
00011
00012 namespace yaodaq
00013 {
00014
00015 enum class MessageType : std::int_least16_t
00016 {
00017   // IXWebSocket MessageType (Message is not set here)
00018   Open     = -1,
00019   Close    = -2,
00020   Error    = -3,
00021   Ping     = -4,
00022   Pong     = -5,
00023   Fragment = -6,
00024   // Unknown should not be used !
00025   Unknown  = 0,
00026 };
00027
00028 inline std::ostream& operator«( std::ostream& os, const MessageType& messageTypes ) { return os «
       static_cast<std::int_least8_t>( messageTypes ); }
00029
00030 }  // namespace yaodaq
00031
00032 #endif  // YAODAQ_MESSAGETYPE
```

9.25 yaodaq/Severity.hpp File Reference

```
#include <cstdint>
```

Namespaces

- namespace yaodaq

Enumerations

- enum class yaodaq::Severity : std::int_least16_t { yaodaq::Info = 1 , yaodaq::Warning = 10 , yaodaq::Error = 100 , yaodaq::Critical = 1000 }

9.26 Severity.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_SEVERITY
00002 #define YAODAQ_SEVERITY
00003
00004 #include <cstdint>
00005
00010 namespace yaodaq
00011 {
00012
00013 enum class Severity : std::int_least16_t
00014 {
00015   Info     = 1,
00016   Warning  = 10,
00017   Error    = 100,
00018   Critical = 1000,
00019 };
00020
00021 }  // namespace yaodaq
00022
00023 #endif  // YAODAQ_SEVERITY
```

9.27 yaodaq/Signal.hpp File Reference

```
#include "yaodaq/Severity.hpp"
#include <cstdint>
```

Namespaces

- namespace yaodaq

Enumerations

- enum class yaodaq::Signal {
  yaodaq::NO = 0 , yaodaq::ABRT = static_cast<int>(Severity::Critical) + 1 , yaodaq::FPE = static_cast<int>(Severity::Critical) + 2 , yaodaq::ILL = static_cast<int>(Severity::Critical) + 3 ,
  yaodaq::SEGV = static_cast<int>(Severity::Critical) + 4 , yaodaq::INT = static_cast<int>(Severity::Warning) + 1 , yaodaq::TERM = static_cast<int>(Severity::Warning) + 2 }

9.28 Signal.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_SIGNAL
00002 #define YAODAQ_SIGNAL
00003
00008 #include "yaodaq/Severity.hpp"
00009
00010 #include <cstdint>
00011
```

```
00012 namespace yaodaq
00013 {
00014
00015 enum class Signal
00016 {
00017   NO   = 0,  // No Signal.
00018   // Critical
00019   ABRT = static_cast<int>( Severity::Critical ) + 1,  // (Signal Abort) Abnormal termination, such as
       is initiated by the abort function.
00020   FPE  = static_cast<int>( Severity::Critical ) + 2,  // (Signal Floating-Point Exception) Erroneous
       arithmetic operation, such as zero divide or an operation resulting in overflow (not necessarily with
       a floating-point operation).
00021   ILL  = static_cast<int>( Severity::Critical ) + 3,  // (Signal Illegal Instruction) Invalid function
       image, such as an illegal instruction. This is generally due to a corruption in the code or to an
       attempt to execute data.
00022   SEGV = static_cast<int>( Severity::Critical ) + 4,  // (Signal Segmentation Violation) Invalid
       access to storage: When a program tries to read or write outside the memory it has allocated.
00023   // Warning
00024   INT  = static_cast<int>( Severity::Warning ) + 1,  // (Signal Interrupt) Interactive attention
       signal. Generally generated by the application user.
00025   TERM = static_cast<int>( Severity::Warning ) + 2,  // (Signal Terminate) Termination request sent to
       program.
00026 };
00027
00028 }  // namespace yaodaq
00029
00030 #endif  // YAODAQ_CLASS_HPP
```

9.29 yaodaq/StatusCode.hpp File Reference

```
#include <cstdint>
```

Namespaces

- namespace yaodaq

Enumerations

- enum class yaodaq::StatusCode : std::int_least32_t { yaodaq::SUCCESS = 0 , yaodaq::LISTEN_ERROR , yaodaq::WRONG_NUMBER_PARAMETERS , yaodaq::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED = 4999 }

9.30 StatusCode.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_STATUSCODE
00002 #define YAODAQ_STATUSCODE
00003
00008 #include <cstdint>
00009
00010 namespace yaodaq
00011 {
00012
00013 enum class StatusCode : std::int_least32_t
00014 {
00015   SUCCESS = 0,
00016   LISTEN_ERROR,
00017   WRONG_NUMBER_PARAMETERS,
00018   CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED = 4999,
00019 };
00020
00021 }  // namespace yaodaq
00022
00023 #endif
```

9.31 yaodaq/Version.hpp File Reference

```
#include <cstdint>
#include <semver.hpp>
```

```
#include <string>
```

Data Structures

- class yaodaq::Version

Namespaces

- namespace yaodaq

9.32 Version.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_VERSION_HPP
00002 #define YAODAQ_VERSION_HPP
00003
00008 #include <cstdint>
00009 #include <semver.hpp>
00010 #include <string>
00011
00012 namespace yaodaq
00013 {
00014
00015 class Version : public semver::version
00016 {
00017 public:
00018   constexpr Version( const std::uint8_t& mj, const std::uint8_t& mn, const std::uint8_t& pt, const
       semver::prerelease& prt = semver::prerelease::none, const std::uint8_t& prn = 0 ) noexcept :
       semver::version( mj, mn, pt, prt, prn ) {}
00019   explicit constexpr Version( const std::string_view& str ) : semver::version( str ) {}
00020   constexpr Version() = default;
00021   std::uint8_t getMajor();
00022   std::uint8_t getMinor();
00023   std::uint8_t getPatch();
00024   std::string  getPreRelease();
00025   std::uint8_t getPreReleaseNumber();
00026 };
00027
00028 }  // namespace yaodaq
00029
00030 #endif  // YAODAQ_VERSION_HPP
```

9.33 yaodaq/WebsocketClient.hpp File Reference

```
#include "yaodaq/Identifier.hpp"
#include "yaodaq/Interrupt.hpp"
#include "yaodaq/LoggerHandler.hpp"
#include "yaodaq/Looper.hpp"
#include <ixwebsocket/IXWebSocket.h>
#include <memory>
#include <spdlog/spdlog.h>
#include <string>
```

Data Structures

- class yaodaq::WebsocketClient

Namespaces

- namespace yaodaq

9.34 WebsocketClient.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_WEBSOCKETCLIENT
00002 #define YAODAQ_WEBSOCKETCLIENT
00003
00008 #include "yaodaq/Identifier.hpp"
00009 #include "yaodaq/Interrupt.hpp"
00010 #include "yaodaq/LoggerHandler.hpp"
00011 #include "yaodaq/Looper.hpp"
00012
00013 #include <ixwebsocket/IXWebSocket.h>
00014 #include <memory>
00015 #include <spdlog/spdlog.h>
00016 #include <string>
00017
00018 namespace yaodaq
00019 {
00020
00021 class Message;
00022 class Open;
00023 class Close;
00024 class Error;
00025 class Ping;
00026 class Pong;
00027 class Fragment;
00028
00029 class WebsocketClient : public ix::WebSocket
00030 {
00031 public:
00032   explicit WebsocketClient( const std::string& name, const std::string& type = "YAODAQWebsocketClient"
      );
00033   virtual ~WebsocketClient();
00034   void                              start();
00035   void                              stop();
00036   void                              loop();
00037   std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
00038
00039   virtual void onMessage( Message& message );
00040   virtual void onOpen( Open& open );
00041   virtual void onClose( Close& close );
00042   virtual void onError( Error& error );
00043   virtual void onPing( Ping& ping );
00044   virtual void onPong( Pong& pong );
00045   virtual void onFragment( Fragment& fragment );
00046
00047 private:
00048   void          onRaisingSignal();
00049   Identifier    m_Identifier;
00050   LoggerHandler m_Logger;
00051   Looper        m_Looper;
00052 };
00053
00054 }  // namespace yaodaq
00055
00056 #endif
```

9.35 yaodaq/WebsocketServer.hpp File Reference

```
#include "yaodaq/Identifier.hpp"
#include "yaodaq/Interrupt.hpp"
#include "yaodaq/LoggerHandler.hpp"
#include "yaodaq/Looper.hpp"
#include <ixwebsocket/IXWebSocketServer.h>
#include <map>
#include <memory>
#include <mutex>
#include <spdlog/spdlog.h>
#include <string>
```

Data Structures

- class yaodaq::WebsocketServer

Namespaces

- namespace yaodaq

9.36 WebsocketServer.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_WEBSOCKETSERVER
00002 #define YAODAQ_WEBSOCKETSERVER
00003
00008 #include "yaodaq/Identifier.hpp"
00009 #include "yaodaq/Interrupt.hpp"
00010 #include "yaodaq/LoggerHandler.hpp"
00011 #include "yaodaq/Looper.hpp"
00012
00013 #include <ixwebsocket/IXWebSocketServer.h>
00014 #include <map>
00015 #include <memory>
00016 #include <mutex>
00017 #include <spdlog/spdlog.h>
00018 #include <string>
00019
00020 namespace yaodaq
00021 {
00022
00023 class Message;
00024 class Open;
00025 class Close;
00026 class Error;
00027 class Ping;
00028 class Pong;
00029 class Fragment;
00030
00031 class WebsocketServer : public ix::WebSocketServer
00032 {
00033 public:
00034   explicit WebsocketServer( const std::string& name, const int& port = ix::SocketServer::kDefaultPort,
       const std::string& host = ix::SocketServer::kDefaultHost, const int& backlog =
       ix::SocketServer::kDefaultTcpBacklog,
00035                             const std::size_t& maxConnections =
       ix::SocketServer::kDefaultMaxConnections, const int& handshakeTimeoutSecs =
       ix::WebSocketServer::kDefaultHandShakeTimeoutSecs, const int& addressFamily =
       ix::SocketServer::kDefaultAddressFamily,
00036                             const std::string& type = "YAODAQWebsocketServer" );
00037   virtual ~WebsocketServer();
00038   void loop();
00039   void start();
00040   void stop( bool useless = true );
00041   void listen();
00042
00043   virtual void onMessage( Message& message );
00044   virtual void onOpen( Open& open );
00045   virtual void onClose( Close& close );
00046   virtual void onError( Error& error );
00047   virtual void onPing( Ping& ping );
00048   virtual void onPong( Pong& pong );
00049   virtual void onFragment( Fragment& fragment );
00050
00051   void setVerbosity( const yaodaq::LoggerHandler::Verbosity& verbosity );
00052
00053   std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
00054
00055   void sendToLoggers( Message& message );
00056   void sendToLoggers( const Message& message );
00057   void sendToLoggers( Message& message, ix::WebSocket& webSocket );
00058   void sendToLoggers( const Message& message, ix::WebSocket& webSocket );
00059
00060 private:
00061   void                                addClient( const Identifier&, ix::WebSocket& );
00062   void                                removeClient( ix::WebSocket& );
00063   void                                onRaisingSignal();
00064   bool                                m_isListening{ false };
00065   Identifier                          m_Identifier;
00066   LoggerHandler                       m_Logger;
00067   Interrupt                           m_Interrupt;
00068   Looper                              m_Looper;
00069   bool                                m_isStopped{ false };
00070   bool                                m_isStarted{ false };
00071   std::map<Identifier, ix::WebSocket&> m_Clients;
00072   std::mutex                          m_Mutex;
00073 };
00074
00075 }  // namespace yaodaq
00076
00077 #endif  // YAODAQ_WEBSOCKETSERVER
```

9.37 yaodaq/ConnectionState.cpp File Reference

```
#include "yaodaq/ConnectionState.hpp"
#include "yaodaq/Identifier.hpp"
```

Namespaces

- namespace yaodaq

9.38 ConnectionState.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/ConnectionState.hpp"
00006
00007 #include "yaodaq/Identifier.hpp"
00008
00009 namespace yaodaq
00010 {
00011
00012 std::list<std::pair<std::string, std::string» ConnectionState::m_Ids{};
00013
00014 ConnectionState::ConnectionState() : ix::ConnectionState() {}
00015
00016 ConnectionState::~ConnectionState()
00017 {
00018   std::lock_guard<std::mutex> guard( m_Mutex );
00019   m_Ids.remove( m_Pair );
00020 }
00021
00022 void ConnectionState::computeId( const std::string& host, const Identifier& id )
00023 {
00024   std::lock_guard<std::mutex> guard( m_Mutex );
00025   m_Pair = std::pair<std::string, std::string>( host, id.getName() );
00026
00027   if( id.empty() ) { _id = std::to_string( _globalId++ ); }
00028   else
00029   {
00030     std::list<std::pair<std::string, std::string»::iterator found = std::find( m_Ids.begin(),
      m_Ids.end(), m_Pair );
00031     if( found == m_Ids.end() )
00032     {
00033       _id = id.getName();
00034       m_Ids.push_back( m_Pair );
00035     }
00036     else
00037     {
00038       setTerminated();
00039     }
00040   }
00041 }
00042
00043 }  // namespace yaodaq
```

9.39 yaodaq/Exception.cpp File Reference

```
#include "yaodaq/Exception.hpp"
```

Namespaces

- namespace yaodaq

9.40 Exception.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Exception.hpp"
00006
00007 namespace yaodaq
```

```
00008 {
00009
00010 std::string Exception::m_Format{ "\n\t[Code] : {Code}\n\t[Description] : {Description}\n\t[File] :
       {File}\n\t[Function] : {Function}\n\t[Line] : {Line}\n\t[Column] : {Column}\n" };
00011
00012 fmt::text_style Exception::m_Style = { fg( fmt::color::crimson ) | fmt::emphasis::bold };
00013
00014 Exception::Exception( const StatusCode& statusCode, const std::string& description, const
       source_location& location ) : source_location( location ), m_Code( static_cast<std::int_least32_t>(
       statusCode ) ), m_Description( description ) { constructMessage(); }
00015
00016 const char* Exception::what() const noexcept { return m_Message.c_str(); }
00017
00018 const char* Exception::description() const noexcept { return m_Description.c_str(); }
00019
00020 std::int_least32_t Exception::code() const noexcept { return m_Code; }
00021
00022 void Exception::constructMessage()
00023 {
00024   m_Message = fmt::format( m_Style, m_Format, fmt::arg( "Code", m_Code ), fmt::arg( "Description",
       m_Description ), fmt::arg( "File", file_name() ), fmt::arg( "Function", function_name() ), fmt::arg(
       "Column", column() ), fmt::arg( "Line", line() ) );
00025 }
00026
00027 }  // namespace yaodaq
```

9.41 yaodaq/Identifier.cpp File Reference

```
#include "yaodaq/Identifier.hpp"
#include "yaodaq/Exception.hpp"
#include "yaodaq/Key.hpp"
#include "yaodaq/StatusCode.hpp"
#include <fmt/color.h>
#include <magic_enum.hpp>
#include <string>
#include <vector>
```

Namespaces

- namespace yaodaq

9.42 Identifier.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Identifier.hpp"
00006
00007 #include "yaodaq/Exception.hpp"
00008 #include "yaodaq/Key.hpp"
00009 #include "yaodaq/StatusCode.hpp"
00010
00011 #include <fmt/color.h>
00012 #include <magic_enum.hpp>
00013 #include <string>
00014 #include <vector>
00015
00016 namespace yaodaq
00017 {
00018
00019 bool Identifier::empty() const
00020 {
00021   if( get() == Identifier().get() ) return true;
00022   else
00023     return false;
00024 }
00025
00026 Identifier::Identifier( const std::string& type, const std::string& name ) : m_Type( type ), m_Name(
       name ) {}
00027
00028 void Identifier::generateKey( const Domain& domain, const Class& c_lass, const Family& family ) {
       m_Key = Key( domain, c_lass, family ); }
00029
00030 std::string Identifier::getDomain() const { return static_cast<std::string>( magic_enum::enum_name(
       magic_enum::enum_cast<Domain>( m_Key.getDomain() ).value() ) ); }
```

```
00031
00032 std::string Identifier::getClass() const { return static_cast<std::string>( magic_enum::enum_name(
       magic_enum::enum_cast<Class>( m_Key.getClass() ).value() ) ); }
00033
00034 std::string Identifier::getFamily() const { return static_cast<std::string>( magic_enum::enum_name(
       magic_enum::enum_cast<Family>( m_Key.getFamily() ).value() ) ); }
00035
00036 std::string Identifier::getType() const { return m_Type; }
00037
00038 std::string Identifier::getName() const { return m_Name; }
00039
00040 Key Identifier::getKey() const { return m_Key; }
00041
00042 std::string Identifier::get() const { return fmt::format( "{0}/{1}/{2}/{3}/{4}", getDomain(),
       getClass(), getFamily(), getType(), getName() ); }
00043
00044 Identifier Identifier::parse( const std::string& id )
00045 {
00046   std::vector<std::string> result;
00047   std::string              tmp        = id;
00048   std::string              separator  = "/";
00049   std::size_t              second_pos = tmp.find( separator );
00050   while( second_pos != std::string::npos )
00051   {
00052     if( 0 != second_pos )
00053     {
00054       std::string word = tmp.substr( 0, second_pos - 0 );
00055       result.push_back( word );
00056     }
00057     else
00058       result.push_back( "" );
00059     tmp        = tmp.substr( second_pos + separator.length() );
00060     second_pos = tmp.find( separator );
00061     if( second_pos == std::string::npos ) result.push_back( tmp );
00062   }
00063   if( result.size() == 5 )
00064   {
00065     Identifier identifier( result[3], result[4] );
00066     identifier.generateKey( magic_enum::enum_cast<Domain>( result[0] ).value(),
       magic_enum::enum_cast<Class>( result[1] ).value(), magic_enum::enum_cast<Family>( result[2] ).value()
       );
00067     return identifier;
00068   }
00069   else
00070   {
00071     throw Exception( StatusCode::WRONG_NUMBER_PARAMETERS, "Number of parameters in key should be 5
       (Domain/Class/Family/Type/Name) !" );
00072   }
00073 }
00074
00075 bool Identifier::operator<( const Identifier& identifier ) const { return this->get() <
       identifier.get(); }
00076
00077 }  // namespace yaodaq
```

9.43 yaodaq/Interrupt.cpp File Reference

```
#include "yaodaq/Interrupt.hpp"
#include "yaodaq/Signal.hpp"
#include <atomic>
#include <csignal>
#include <mutex>
#include <thread>
```

Namespaces

- namespace yaodaq

9.44 Interrupt.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Interrupt.hpp"
00006
00007 #include "yaodaq/Signal.hpp"
```

```
00008
00009 #include <atomic>
00010 #include <csignal>
00011 #include <mutex>
00012 #include <thread>
00013
00014 namespace yaodaq
00015 {
00016
00017 volatile std::atomic<Signal> Interrupt::m_Signal = Signal::NO;
00018
00019 Interrupt::Interrupt() { init(); }
00020
00021 void Interrupt::restore()
00022 {
00023   std::signal( SIGTERM, SIG_DFL );
00024   std::signal( SIGSEGV, SIG_DFL );
00025   std::signal( SIGINT, SIG_DFL );
00026   std::signal( SIGILL, SIG_DFL );
00027   std::signal( SIGABRT, SIG_DFL );
00028   std::signal( SIGFPE, SIG_DFL );
00029 }
00030
00031 void Interrupt::init()
00032 {
00033   setSignal( Signal::TERM );
00034   setSignal( Signal::TERM );
00035   setSignal( Signal::SEGV );
00036   setSignal( Signal::INT );
00037   setSignal( Signal::ILL );
00038   setSignal( Signal::ABRT );
00039   setSignal( Signal::FPE );
00040 }
00041
00042 Interrupt::~Interrupt() { restore(); }
00043
00044 Signal Interrupt::getSignal()
00045 {
00046   if( m_Signal.load() != Signal::NO )
00047   {
00048     std::lock_guard<std::mutex> guard( m_mutex );
00049     init();
00050   }
00051   return m_Signal.load();
00052 }
00053
00054 void Interrupt::setSignal( const Signal& signal )
00055 {
00056   switch( signal )
00057   {
00058     case Signal::ABRT: std::signal( SIGABRT, []( int ) -> void { m_Signal.store( Signal::ABRT ); } );
      break;
00059     case Signal::FPE: std::signal( SIGFPE, []( int ) -> void { m_Signal.store( Signal::FPE ); } );
      break;
00060     case Signal::ILL: std::signal( SIGILL, []( int ) -> void { m_Signal.store( Signal::ILL ); } );
      break;
00061     case Signal::SEGV: std::signal( SIGSEGV, []( int ) -> void { m_Signal.store( Signal::SEGV ); } );
      break;
00062     case Signal::INT: std::signal( SIGINT, []( int ) -> void { m_Signal.store( Signal::INT ); } );
      break;
00063     case Signal::TERM: std::signal( SIGTERM, []( int ) -> void { m_Signal.store( Signal::TERM ); } );
      break;
00064     default: break;
00065   }
00066 }
00067
00068 }  // namespace yaodaq
```

9.45 yaodaq/IXWebsocketMessage.cpp File Reference

`#include "yaodaq/IXWebsocketMessage.hpp"`

Namespaces

- namespace yaodaq

9.46 IXWebsocketMessage.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/IXWebsocketMessage.hpp"
00006
00007 namespace yaodaq
00008 {
00009
00010 IXMessage::IXMessage( const MessageType& messageType ) : Message( messageType ) {}
00011
00012 void IXMessage::setConnectionStateInfos( std::shared_ptr<ConnectionState>& connectionState )
00013 {
00014   nlohmann::json j = getContent();
00015   j["id"]          = connectionState->getId();
00016   j["remote_ip"]   = connectionState->getRemoteIp();
00017   j["remote_port"] = connectionState->getRemotePort();
00018   setContent( j );
00019 }
00020
00021 // Open
00022 Open::Open( const ix::WebSocketOpenInfo& openInfo ) : IXMessage( MessageType::Open )
00023 {
00024   nlohmann::json j = getContent();
00025   j["uri"]         = openInfo.uri;
00026   j["headers"]     = openInfo.headers;
00027   j["protocol"]    = openInfo.protocol;
00028   setContent( j );
00029 }
00030
00031 Open::Open( const ix::WebSocketOpenInfo& openInfo, std::shared_ptr<ConnectionState>& connectionState )
       : Open( openInfo ) { setConnectionStateInfos( connectionState ); }
00032
00033 std::string Open::getURI() const { return get()["content"]["uri"].get<std::string>(); }
00034
00035 std::map<std::string, std::string> Open::getHeaders() const
00036 {
00037   std::map<std::string, std::string> ret = get()["content"]["headers"].get<std::map<std::string,
       std::string>>();
00038   return ret;
00039 }
00040
00041 std::string Open::getProtocol() const { return get()["content"]["protocol"].get<std::string>(); }
00042
00043 // Close
00044 Close::Close( const ix::WebSocketCloseInfo& closeInfo ) : IXMessage( MessageType::Close )
00045 {
00046   nlohmann::json j;
00047   j["code"]   = closeInfo.code;
00048   j["reason"] = closeInfo.reason;
00049   j["remote"] = closeInfo.remote;
00050   setContent( j );
00051 }
00052
00053 Close::Close( const ix::WebSocketCloseInfo& closeInfo, std::shared_ptr<ConnectionState>&
       connectionState ) : Close( closeInfo ) { setConnectionStateInfos( connectionState ); }
00054
00055 std::uint16_t Close::getCode() const { return get()["content"]["code"].get<std::uint16_t>(); }
00056 std::string   Close::getReason() const { return get()["content"]["reason"].get<std::string>(); }
00057 bool          Close::getRemote() const { return get()["content"]["remote"].get<bool>(); }
00058
00059 // Error
00060 Error::Error( const ix::WebSocketErrorInfo& errorInfo ) : IXMessage( MessageType::Error )
00061 {
00062   nlohmann::json j;
00063   j["retries"]             = errorInfo.retries;
00064   j["wait_time"]           = errorInfo.wait_time;
00065   j["http_status"]         = errorInfo.http_status;
00066   j["reason"]              = errorInfo.reason;
00067   j["decompression_error"] = errorInfo.decompressionError;
00068   setContent( j );
00069 }
00070
00071 Error::Error( const ix::WebSocketErrorInfo& errorInfo, std::shared_ptr<ConnectionState>&
       connectionState ) : Error( errorInfo ) { setConnectionStateInfos( connectionState ); }
00072
00073 std::uint16_t Error::getRetries() const { return get()["content"]["retries"].get<std::uint16_t>(); }
00074
00075 double Error::getWaitTime() const { return get()["content"]["wait_time"].get<double>(); }
00076
00077 int Error::getHttpStatus() const { return get()["content"]["http_status"].get<int>(); }
00078
00079 std::string Error::getReason() const { return get()["content"]["reason"].get<std::string>(); }
00080
00081 bool Error::getDecompressionError() const { return
       get()["content"]["decompression_error"].get<bool>(); }
```

```
00082
00083 // Ping
00084 Ping::Ping( const ix::WebSocketMessagePtr& ping ) : IXMessage( MessageType::Ping ) {}
00085
00086 Ping::Ping( const ix::WebSocketMessagePtr& ping, std::shared_ptr<ConnectionState>& connectionState ) :
       Ping( ping ) { setConnectionStateInfos( connectionState ); }
00087
00088 // Pong
00089 Pong::Pong( const ix::WebSocketMessagePtr& pong ) : IXMessage( MessageType::Pong ) {}
00090
00091 Pong::Pong( const ix::WebSocketMessagePtr& pong, std::shared_ptr<ConnectionState>& connectionState ) :
       Pong( pong ) { setConnectionStateInfos( connectionState ); }
00092
00093 // Fragment
00094 Fragment::Fragment( const ix::WebSocketMessagePtr& fragment ) : IXMessage( MessageType::Fragment ) {}
00095
00096 Fragment::Fragment( const ix::WebSocketMessagePtr& fragment, std::shared_ptr<ConnectionState>&
       connectionState ) : Fragment( fragment ) { setConnectionStateInfos( connectionState ); }
00097
00098 }  // namespace yaodaq
```

9.47 yaodaq/Key.cpp File Reference

```
#include "yaodaq/Key.hpp"
#include <cstdint>
```

Namespaces

- namespace yaodaq

9.48 Key.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Key.hpp"
00006
00007 #include <cstdint>
00008
00009 namespace yaodaq
00010 {
00011 Key::Key( const Domain& domain, const Class& c_lass, const Family& family ) { m_Key = (
       static_cast<std::int_least8_t>( domain ) « 24 ) + ( static_cast<std::int_least8_t>( c_lass ) « 16 ) +
       static_cast<std::int_least16_t>( family ); }
00012
00013 std::int_least8_t Key::getDomain() const { return ( m_Key » 24 ) & 0xFF; }
00014
00015 std::int_least8_t Key::getClass() const { return ( m_Key » 16 ) & 0xFF; }
00016
00017 std::int_least16_t Key::getFamily() const { return (m_Key)&0xFFFF; }
00018
00019 std::int_least32_t Key::getKey() const { return m_Key; }
00020
00021 }  // namespace yaodaq
```

9.49 yaodaq/LoggerHandler.cpp File Reference

```
#include "yaodaq/LoggerHandler.hpp"
#include "spdlog/spdlog.h"
```

Namespaces

- namespace yaodaq

9.50 LoggerHandler.cpp

Go to the documentation of this file.

```
00001
```

```
00005 #include "yaodaq/LoggerHandler.hpp"
00006
00007 #include "spdlog/spdlog.h"
00008
00009 namespace yaodaq
00010 {
00011
00012 LoggerHandler::LoggerHandler() { init(); }
00013
00014 void LoggerHandler::setName( const std::string& name )
00015 {
00016   m_Name = name;
00017   init();
00018 }
00019
00020 LoggerHandler::~LoggerHandler() {}
00021
00022 void LoggerHandler::setVerbosity( const Verbosity& verbosity )
00023 {
00024   m_Verbosity = verbosity;
00025   init();
00026 }
00027
00028 void LoggerHandler::init()
00029 {
00030   m_Logger = std::make_shared<spdlog::logger>( m_Name, std::begin( m_Sinks ), std::end( m_Sinks ) );
00031   switch( m_Verbosity )
00032   {
00033     case Verbosity::Off: m_Logger->set_level( spdlog::level::off ); break;
00034     case Verbosity::Trace: m_Logger->set_level( spdlog::level::trace ); break;
00035     case Verbosity::Debug: m_Logger->set_level( spdlog::level::debug ); break;
00036     case Verbosity::Info: m_Logger->set_level( spdlog::level::info ); break;
00037     case Verbosity::Warn: m_Logger->set_level( spdlog::level::warn ); break;
00038     case Verbosity::Error: m_Logger->set_level( spdlog::level::err ); break;
00039     case Verbosity::Critical: m_Logger->set_level( spdlog::level::critical ); break;
00040   }
00041 }
00042
00043 std::shared_ptr<spdlog::logger> LoggerHandler::logger() { return std::shared_ptr<spdlog::logger>(
      m_Logger ); }
00044
00045 void LoggerHandler::addSink( const spdlog::sink_ptr& sink )
00046 {
00047   m_Sinks.push_back( sink );
00048   init();
00049 }
00050
00051 void LoggerHandler::clearSinks()
00052 {
00053   m_Sinks.clear();
00054   init();
00055 }
00056
00057 }  // namespace yaodaq
```

9.51 yaodaq/Looper.cpp File Reference

```
#include "yaodaq/Looper.hpp"
#include <chrono>
#include <thread>
```

Namespaces

- namespace yaodaq

9.52 Looper.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Looper.hpp"
00006
00007 #include <chrono>
00008 #include <thread>
00009
00010 namespace yaodaq
00011 {
```

```
00012
00013 int Looper::m_instance{ 0 };
00014
00015 Interrupt Looper::m_Interrupt{ Interrupt{} };
00016
00017 int Looper::getInstance() { return m_instance; }
00018
00019 void Looper::supressInstance()
00020 {
00021   if( m_hasBeenSupressed == false )
00022   {
00023     m_hasBeenSupressed = true;
00024     m_instance--;
00025   }
00026 }
00027
00028 Looper::Looper()
00029 {
00030   if( m_hasBeenAdded == false )
00031   {
00032     m_hasBeenAdded = true;
00033     ++m_instance;
00034   }
00035 }
00036
00037 Signal Looper::loop()
00038 {
00039   static Signal signal{ yaodaq::Signal::NO };
00040   if( m_instance == 0 )
00041   {
00042     do {
00043       signal = m_Interrupt.getSignal();
00044       std::this_thread::sleep_for( std::chrono::microseconds( 1 ) );
00045     } while( signal == yaodaq::Signal::NO );
00046   }
00047   return signal;
00048 }
00049
00050 Signal Looper::getSignal() { return m_Interrupt.getSignal(); }
00051
00052 Looper::~Looper()
00053 {
00054   if( m_hasBeenAdded == true && m_hasBeenSupressed == false )
00055   {
00056     m_hasBeenSupressed = true;
00057     --m_instance;
00058   }
00059 }
00060
00061 }  // namespace yaodaq
```

9.53 yaodaq/Message.cpp File Reference

```
#include "yaodaq/Message.hpp"
#include "fmt/chrono.h"
#include "magic_enum.hpp"
#include "yaodaq/Classification.hpp"
#include "yaodaq/Identifier.hpp"
#include <chrono>
#include <ctime>
#include <ixwebsocket/IXUuid.h>
#include <string>
#include <ixwebsocket/IXWebSocketVersion.h>
#include <yaodaq/YaodaqVersion.hpp>
```

Namespaces

- namespace yaodaq

9.54 Message.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Message.hpp"
00006
00007 #include "fmt/chrono.h"
00008 #include "magic_enum.hpp"
00009 #include "yaodaq/Classification.hpp"
00010 #include "yaodaq/Identifier.hpp"
00011
00012 #include <chrono>
00013 #include <ctime>
00014 #include <ixwebsocket/IXUuid.h>
00015 #include <string>
00016
00017 // Versions numbers
00018 #include <ixwebsocket/IXWebSocketVersion.h>
00019 #include <yaodaq/YaodaqVersion.hpp>
00020
00021 namespace yaodaq
00022 {
00023
00024 Message::Message()
00025 {
00026   m_JSON["from"];
00027   m_JSON["to"];
00028   m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
00029   // m_JSON["uuid"] = ix::uuid4();
00030   m_JSON["content"];
00031   m_JSON["timestamp"] = fmt::format( "{:%F %T %z}", fmt::gmtime( std::chrono::system_clock::to_time_t(
       std::chrono::system_clock::now() ) ) );
00032   // m_JSON["meta"]["compiler"]                = nlohmann::json::meta()["compiler"];
00033   // m_JSON["meta"]["platform"]                = nlohmann::json::meta()["platform"];
00034   // m_JSON["meta"]["versions"]["json"]        = nlohmann::json::meta()["version"]["string"];
00035   // m_JSON["meta"]["versions"]["yaodaq"]      = yaodaq_version.to_string();
00036   // m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
00037 }
00038
00039 void Message::setContent( const nlohmann::json& content ) { m_JSON["content"] =
       static_cast<nlohmann::json>( content ); }
00040
00041 void Message::setContent( const std::string& content )
00042 {
00043   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00044   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00045 }
00046
00047 void Message::setContent( const char* content )
00048 {
00049   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00050   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00051 }
00052
00053 Message::Message( const nlohmann::json& content, const MessageType& messageType ) : Message(
       messageType ) { setContent( content ); }
00054
00055 Message::Message( const std::string& content, const MessageType& messageType ) : Message( messageType
       ) { setContent( content ); }
00056
00057 Message::Message( const char* content, const MessageType& messageType ) : Message( messageType ) {
       setContent( content ); }
00058
00059 std::string Message::dump( const int& indent, const char& indent_char, const bool& ensure_ascii, const
       nlohmann::detail::error_handler_t& error_handler ) const { return m_JSON.dump( indent, indent_char,
       ensure_ascii, error_handler ); }
00060
00061 nlohmann::json Message::get() const { return m_JSON; }
00062
00063 std::string Message::getTypeName() const { return m_JSON["type"].get<std::string>(); }
00064
00065 MessageType Message::getTypeValue() const { return magic_enum::enum_cast<MessageType>(
       m_JSON["type"].get<std::string>() ).value(); }
00066
00067 nlohmann::json Message::getContent() const { return m_JSON["content"]; }
00068
00069 std::string Message::getTimestamp() const { return m_JSON["timestamp"].get<std::string>(); }
00070
00071 std::time_t Message::getTime() const
00072 {
00073   std::tm tm;
00074   memset( &tm, 0, sizeof( tm ) );
00075   std::istringstream ss( getTimestamp() );
00076   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00077   return mktime( &tm );
00078 }
00079
00080 void Message::setFrom( const Identifier& identifier )
00081 {
00082   m_JSON["from"]["name"]   = identifier.getName();
```

```
00083   m_JSON["from"]["type"]   = identifier.getType();
00084   m_JSON["from"]["family"] = identifier.getFamily();
00085   m_JSON["from"]["class"]  = identifier.getClass();
00086   m_JSON["from"]["domain"] = identifier.getDomain();
00087 }
00088
00089 Identifier Message::getIdentifier() const
00090 {
00091   if( m_JSON["from"].is_null() ) return {};
00092   else
00093   {
00094     Identifier id( m_JSON["from"]["type"].get<std::string>(),
      m_JSON["from"]["name"].get<std::string>() );
00095     id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
      ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00096                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
      ).value() );
00097     return id;
00098   }
00099 }
00100
00101 Message::Message( const MessageType& messageType ) : Message() { m_JSON["type"] =
      magic_enum::enum_name( messageType ); }
00102
00103 }  // namespace yaodaq
```

9.55 yaodaq/Version.cpp File Reference

```
#include "yaodaq/Version.hpp"
#include <magic_enum.hpp>
```

Namespaces

- namespace yaodaq

9.56 Version.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Version.hpp"
00006
00007 #include <magic_enum.hpp>
00008
00009 namespace yaodaq
00010 {
00011
00012 std::uint8_t Version::getMajor() { return major; }
00013
00014 std::uint8_t Version::getMinor() { return minor; }
00015
00016 std::uint8_t Version::getPatch() { return patch; }
00017
00018 std::string Version::getPreRelease() { return std::string( magic_enum::enum_name( prerelease_type ) );
      }
00019
00020 std::uint8_t Version::getPreReleaseNumber() { return prerelease_number; }
00021
00022 const static Version yaodaq_version;
00023
00024 }  // namespace yaodaq
```

9.57 yaodaq/WebsocketClient.cpp File Reference

```
#include "yaodaq/WebsocketClient.hpp"
#include "yaodaq/Exception.hpp"
#include "yaodaq/IXWebsocketMessage.hpp"
#include "yaodaq/StatusCode.hpp"
#include <chrono>
#include <ixwebsocket/IXNetSystem.h>
#include <magic_enum.hpp>
```

```
#include <spdlog/sinks/stdout_color_sinks.h>
#include <thread>
```

Namespaces

- namespace yaodaq

9.58 WebsocketClient.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/WebsocketClient.hpp"
00006
00007 #include "yaodaq/Exception.hpp"
00008 #include "yaodaq/IXWebsocketMessage.hpp"
00009 #include "yaodaq/StatusCode.hpp"
00010
00011 #include <chrono>
00012 #include <ixwebsocket/IXNetSystem.h>
00013 #include <magic_enum.hpp>
00014 #include <spdlog/sinks/stdout_color_sinks.h>
00015 #include <thread>
00016
00017 namespace yaodaq
00018 {
00019
00020 WebsocketClient::WebsocketClient( const std::string& name, const std::string& type ) : m_Identifier(
      type, name )
00021 {
00022   ix::initNetSystem();
00023
00024   m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
00025   m_Logger.setName( m_Identifier.get() );
00026   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00027
00028   ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
00029   setExtraHeaders( header );
00030
00031   setOnMessageCallback(
00032     [this]( const ix::WebSocketMessagePtr& msg )
00033     {
00034       if( msg->type == ix::WebSocketMessageType::Message ) { logger()->error( "{}", msg->str ); }
00035       else if( msg->type == ix::WebSocketMessageType::Error )
00036       {
00037         std::cout « "Connection error: " « msg->errorInfo.reason « std::endl;
00038       }
00039       else if( msg->type == ix::WebSocketMessageType::Close )
00040       {
00041         disableAutomaticReconnection();
00042         if( msg->closeInfo.code == magic_enum::enum_integer(
      StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ) )
00043         {
00044           logger()->critical( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold,
      msg->closeInfo.reason ) );
00045           close();
00046           // throw Exception( StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED,
      msg->closeInfo.reason );
00047         }
00048       }
00049     }
00050
00051   );
00052 }
00053
00054 void WebsocketClient::onMessage( Message& message ) {}
00055
00056 void WebsocketClient::onOpen( Open& open ) {}
00057
00058 void WebsocketClient::onClose( Close& close ) {}
00059
00060 void WebsocketClient::onError( Error& error ) {}
00061
00062 void WebsocketClient::onPing( Ping& ping ) {}
00063
00064 void WebsocketClient::onPong( Pong& pong ) {}
00065
00066 void WebsocketClient::onFragment( Fragment& fragment ) {}
00067
00068 WebsocketClient::~WebsocketClient()
00069 {
```

```
00070   stop();
00071   ix::uninitNetSystem();
00072 }
00073
00074 void WebsocketClient::start()
00075 {
00076   if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00077   {
00078     logger()->trace( "Client started. Connected to {}", getUrl() );
00079     ix::WebSocket::start();
00080   }
00081 }
00082
00083 void WebsocketClient::stop()
00084 {
00085   if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00086   {
00087     logger()->trace( "Client stopped" );
00088     ix::WebSocket::stop();
00089     while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
      std::chrono::microseconds( 1 ) ); }
00090   }
00091 }
00092
00093 void WebsocketClient::loop()
00094 {
00095   WebsocketClient::start();
00096   m_Looper.supressInstance();
00097   onRaisingSignal();
00098 }
00099
00100 void WebsocketClient::onRaisingSignal()
00101 {
00102   Signal signal = m_Looper.loop();
00103   if( m_Looper.getInstance() == 0 )
00104   {
00105     int value = magic_enum::enum_integer( signal );
00106     if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
      "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00107     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
00108     {
00109       logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00110     }
00111     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
00112     {
00113       fmt::print( "\n" );
00114       logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00115     }
00116     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
00117     {
00118       fmt::print( "\n" );
00119       logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00120     }
00121     else
00122     {
00123       fmt::print( "\n" );
00124       logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00125     }
00126     if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
      std::exit( magic_enum::enum_integer( signal ) );
00127   }
00128 }
00129
00130 }  // namespace yaodaq
```

9.59 yaodaq/WebsocketServer.cpp File Reference

```
#include "yaodaq/WebsocketServer.hpp"
#include "yaodaq/Classification.hpp"
#include "yaodaq/ConnectionState.hpp"
#include "yaodaq/Exception.hpp"
#include "yaodaq/IXWebsocketMessage.hpp"
#include "yaodaq/Identifier.hpp"
#include "yaodaq/StatusCode.hpp"
#include <chrono>
#include <iostream>
#include <ixwebsocket/IXNetSystem.h>
#include <magic_enum.hpp>
#include <spdlog/sinks/stdout_color_sinks.h>
```

```
#include <spdlog/spdlog.h>
#include <string>
#include <thread>
#include <utility>
```

Namespaces

- namespace yaodaq

9.60 WebsocketServer.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/WebsocketServer.hpp"
00006
00007 #include "yaodaq/Classification.hpp"
00008 #include "yaodaq/ConnectionState.hpp"
00009 #include "yaodaq/Exception.hpp"
00010 #include "yaodaq/IXWebsocketMessage.hpp"
00011 #include "yaodaq/Identifier.hpp"
00012 #include "yaodaq/StatusCode.hpp"
00013
00014 #include <chrono>
00015 #include <iostream>
00016 #include <ixwebsocket/IXNetSystem.h>
00017 #include <magic_enum.hpp>
00018 #include <spdlog/sinks/stdout_color_sinks.h>
00019 #include <spdlog/spdlog.h>
00020 #include <string>
00021 #include <thread>
00022 #include <utility>
00023
00024 namespace yaodaq
00025 {
00026
00027 WebsocketServer::WebsocketServer( const std::string& name, const int& port, const std::string& host,
       const int& backlog, const std::size_t& maxConnections, const int& handshakeTimeoutSecs, const int&
       addressFamily, const std::string& type ) :
00028   ix::WebSocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
       m_Identifier( type, name )
00029 {
00030   ix::initNetSystem();
00031
00032   m_Identifier.generateKey( Domain::Application, Class::Server, Family::WebSocketServer );
00033   m_Logger.setName( m_Identifier.get() );
00034   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00035
00036   setConnectionStateFactory( []() { return std::make_shared<ConnectionState>(); } );
00037
00038   setOnClientMessageCallback(
00039     [this]( std::shared_ptr<ix::ConnectionState> connectionState, ix::WebSocket& webSocket, const
       ix::WebSocketMessagePtr& msg )
00040     {
00041       // The ConnectionState object contains information about the connection
00042       std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>(
       connectionState );
00043       if( msg->type == ix::WebSocketMessageType::Message ) {}
00044       else if( msg->type == ix::WebSocketMessageType::Open )
00045       {
00046         // Check if a client with the same name is already connected;
00047         connection->computeId( getHost() + ":" + std::to_string( getPort() ), Identifier::parse(
       msg->openInfo.headers["id"] ) );
00048         if( connection->isTerminated() )
00049         {
00050           logger()->error( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold, "One client with
       the name \"{}\" is already connected !", Identifier::parse( msg->openInfo.headers["id"] ).getName() )
       );
00051           webSocket.stop( magic_enum::enum_integer(
       StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ),
00052                           fmt::format( "One client with the name \"{}\" is already connected to
       ws{}://{}:{} !", Identifier::parse( msg->openInfo.headers["id"] ).getName(), "", getHost(), getPort()
       ) );
00053           std::this_thread::sleep_for( std::chrono::milliseconds( 10 ) );
00054           return;
00055         }
00056         addClient( Identifier::parse( msg->openInfo.headers["id"] ), webSocket );
00057         Open open( msg->openInfo, connection );
00058         sendToLoggers( open, webSocket );
00059         onOpen( open );
```

```
00060       }
00061       else if( msg->type == ix::WebSocketMessageType::Close )
00062       {
00063         Close close( msg->closeInfo, connection );
00064         sendToLoggers( close, webSocket );
00065         onClose( close );
00066         removeClient( webSocket );
00067       }
00068       else if( msg->type == ix::WebSocketMessageType::Error )
00069       {
00070         Error error( msg->errorInfo, connection );
00071         sendToLoggers( error, webSocket );
00072         onError( error );
00073       }
00074       else if( msg->type == ix::WebSocketMessageType::Ping )
00075       {
00076         Ping ping( msg, connection );
00077         sendToLoggers( ping, webSocket );
00078         onPing( ping );
00079       }
00080       else if( msg->type == ix::WebSocketMessageType::Pong )
00081       {
00082         Pong pong( msg, connection );
00083         sendToLoggers( pong, webSocket );
00084         onPong( pong );
00085       }
00086       else if( msg->type == ix::WebSocketMessageType::Fragment )
00087       {
00088         Fragment fragment( msg, connection );
00089         sendToLoggers( fragment, webSocket );
00090         onFragment( fragment );
00091       }
00092     } );
00093 }
00094
00095 void WebsocketServer::addClient( const Identifier& identifier, ix::WebSocket& websocket )
00096 {
00097   std::lock_guard<std::mutex> guard( m_Mutex );
00098   m_Clients.try_emplace( identifier, websocket );
00099 }
00100
00101 void WebsocketServer::removeClient( ix::WebSocket& websocket )
00102 {
00103   std::lock_guard<std::mutex> guard( m_Mutex );
00104   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00105   {
00106     if( &it->second == &websocket )
00107     {
00108       m_Clients.erase( it->first );
00109       break;
00110     }
00111   }
00112 }
00113
00114 void WebsocketServer::sendToLoggers( Message& message, ix::WebSocket& webSocket )
00115 {
00116   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00117   {
00118     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
      != &it->second ) it->second.send( message.dump() );
00119   }
00120 }
00121
00122 void WebsocketServer::sendToLoggers( Message& message )
00123 {
00124   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00125   {
00126     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
      it->second.send( message.dump() );
00127   }
00128 }
00129
00130 void WebsocketServer::sendToLoggers( const Message& message, ix::WebSocket& webSocket )
00131 {
00132   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00133   {
00134     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
      != &it->second ) it->second.send( message.dump() );
00135   }
00136 }
00137
00138 void WebsocketServer::sendToLoggers( const Message& message )
00139 {
```

```
00140   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
       ++it )
00141   {
00142     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
       it->second.send( message.dump() );
00143   }
00144 }
00145
00146 void WebsocketServer::onMessage( Message& message ) {}
00147
00148 void WebsocketServer::onOpen( Open& open ) {}
00149
00150 void WebsocketServer::onClose( Close& close ) {}
00151
00152 void WebsocketServer::onError( Error& error ) {}
00153
00154 void WebsocketServer::onPing( Ping& ping ) {}
00155
00156 void WebsocketServer::onPong( Pong& pong ) {}
00157
00158 void WebsocketServer::onFragment( Fragment& fragment ) {}
00159
00160 void WebsocketServer::listen()
00161 {
00162   if( !m_isListening )
00163   {
00164     std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
00165     if( ret.first )
00166     {
00167       m_isListening = ret.first;
00168       logger()->info( "Server listening on {0}:{1}", getHost(), getPort() );
00169     }
00170     else
00171       throw Exception( StatusCode::LISTEN_ERROR, ret.second );
00172   }
00173 }
00174
00175 void WebsocketServer::start()
00176 {
00177   if( !m_isStarted )
00178   {
00179     m_isStarted = true;
00180     logger()->trace( "Server started" );
00181     ix::WebSocketServer::start();
00182   }
00183 }
00184
00185 void WebsocketServer::stop( bool useless )
00186 {
00187   if( !m_isStopped )
00188   {
00189     m_isStopped = true;
00190     useless     = !useless;
00191     logger()->trace( "Server stopped" );
00192     ix::WebSocketServer::stop();
00193   }
00194 }
00195
00196 void WebsocketServer::setVerbosity( const yaodaq::LoggerHandler::Verbosity& verbosity ) {
       m_Logger.setVerbosity( verbosity ); }
00197
00198 WebsocketServer::~WebsocketServer()
00199 {
00200   stop();
00201   ix::uninitNetSystem();
00202 }
00203
00204 void WebsocketServer::loop()
00205 {
00206   listen();
00207   start();
00208   m_Looper.supressInstance();
00209   onRaisingSignal();
00210 }
00211
00212 void WebsocketServer::onRaisingSignal()
00213 {
00214   Signal signal = m_Looper.loop();
00215   if( m_Looper.getInstance() == 0 )
00216   {
00217     int value = magic_enum::enum_integer( signal );
00218     if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
       "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00219     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
00220     {
00221       logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00222     }
```

```
00223     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
00224     {
00225       fmt::print( "\n" );
00226       logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00227     }
00228     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
00229     {
00230       fmt::print( "\n" );
00231       logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00232     }
00233     else
00234     {
00235       fmt::print( "\n" );
00236       logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00237     }
00238     if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
      std::exit( magic_enum::enum_integer( signal ) );
00239   }
00240 }
00241
00242 }  // namespace yaodaq
```