



# YAODAQ

Y e t   A n   O t h e r   D A Q

Generated by Doxygen 1.9.2



<b>1 License</b>	<b>1</b>
<b>2 Third-party licenses</b>	<b>3</b>
2.1 LICENSE ISSUES	4
2.1.1 OpenSSL License	4
2.1.2 Original SSLeay License	6
<b>3 Namespace Index</b>	<b>11</b>
3.1 Namespace List	11
<b>4 Hierarchical Index</b>	<b>13</b>
4.1 Class Hierarchy	13
<b>5 Data Structure Index</b>	<b>15</b>
5.1 Data Structures	15
<b>6 File Index</b>	<b>17</b>
6.1 File List	17
<b>7 Namespace Documentation</b>	<b>19</b>
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 yaodag 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	20
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
<b>8 Data Structure Documentation</b>	<b>25</b>
8.1 yaodag::ConnectionState Class Reference	25
8.1.1 Detailed Description	25
8.1.2 Constructor & Destructor Documentation	25
8.1.2.1 ConnectionState()	25
8.1.2.2 ~ConnectionState()	25
8.1.3 Member Function Documentation	25
8.1.3.1 computeId()	26

8.2 yaodaq::Exception Class Reference	26
8.2.1 Detailed Description	26
8.2.2 Constructor & Destructor Documentation	26
8.2.2.1 Exception() [1/2]	27
8.2.2.2 Exception() [2/2]	27
8.2.2.3 ~Exception()	27
8.2.3 Member Function Documentation	27
8.2.3.1 code()	27
8.2.3.2 description()	27
8.2.3.3 setFormat()	27
8.2.3.4 setStyle()	27
8.2.3.5 what()	27
8.3 yaodaq::Identifier Class Reference	28
8.3.1 Detailed Description	28
8.3.2 Constructor & Destructor Documentation	28
8.3.2.1 Identifier() [1/2]	28
8.3.2.2 Identifier() [2/2]	28
8.3.3 Member Function Documentation	28
8.3.3.1 empty()	28
8.3.3.2 generateKey()	29
8.3.3.3 get()	29
8.3.3.4 getClass()	29
8.3.3.5 getDomain()	29
8.3.3.6 getFamily()	29
8.3.3.7 getKey()	29
8.3.3.8 getName()	29
8.3.3.9 getType()	29
8.3.3.10 parse()	30
8.4 yaodaq::Interrupt Class Reference	30
8.4.1 Detailed Description	30
8.4.2 Constructor & Destructor Documentation	30
8.4.2.1 Interrupt()	30
8.4.2.2 ~Interrupt()	31
8.4.3 Member Function Documentation	31
8.4.3.1 getSignal()	31
8.4.3.2 init()	31
8.4.3.3 restore()	31
8.5 yaodaq::Key Class Reference	31
8.5.1 Detailed Description	32
8.5.2 Constructor & Destructor Documentation	32
8.5.2.1 Key() [1/2]	32
8.5.2.2 Key() [2/2]	32

8.5.3 Member Function Documentation	32
8.5.3.1 getClass()	32
8.5.3.2 getDomain()	32
8.5.3.3 getFamily()	32
8.5.3.4 getKey()	32
8.6 yaodag::LoggerHandler Class Reference	32
8.6.1 Detailed Description	33
8.6.2 Member Enumeration Documentation	33
8.6.2.1 Verbosity	33
8.6.3 Constructor & Destructor Documentation	33
8.6.3.1 LoggerHandler()	33
8.6.3.2 ~LoggerHandler()	34
8.6.4 Member Function Documentation	34
8.6.4.1 addSink()	34
8.6.4.2 clearSinks()	34
8.6.4.3 logger()	34
8.6.4.4 setName()	34
8.6.4.5 setVerbosity()	34
8.7 yaodag::Looper Class Reference	34
8.7.1 Detailed Description	35
8.7.2 Constructor & Destructor Documentation	35
8.7.2.1 Looper()	35
8.7.2.2 ~Looper()	35
8.7.3 Member Function Documentation	35
8.7.3.1 getInstance()	35
8.7.3.2 getSignal()	35
8.7.3.3 loop()	36
8.7.3.4 supressInstance()	36
8.8 yaodag::Message Class Reference	36
8.8.1 Detailed Description	37
8.8.2 Constructor & Destructor Documentation	37
8.8.2.1 Message() [1/5]	37
8.8.2.2 Message() [2/5]	37
8.8.2.3 Message() [3/5]	37
8.8.2.4 Message() [4/5]	37
8.8.2.5 Message() [5/5]	38
8.8.3 Member Function Documentation	38
8.8.3.1 dump()	38
8.8.3.2 get()	38
8.8.3.3 getContent()	38
8.8.3.4 getIdentifier()	38
8.8.3.5 getTime()	38

8.8.3.6	getTimestamp()	39
8.8.3.7	getTypeName()	39
8.8.3.8	getTypeValue()	39
8.8.3.9	setContent() [1/3]	39
8.8.3.10	setContent() [2/3]	39
8.8.3.11	setContent() [3/3]	39
8.8.3.12	setFrom()	39
8.9	yaodag::Open Class Reference	40
8.9.1	Detailed Description	40
8.9.2	Constructor & Destructor Documentation	40
8.9.2.1	Open() [1/2]	41
8.9.2.2	Open() [2/2]	41
8.9.3	Member Function Documentation	41
8.9.3.1	dump()	41
8.9.3.2	get()	41
8.9.3.3	getContent()	41
8.9.3.4	getHeaders()	42
8.9.3.5	getIdentifier()	42
8.9.3.6	getProtocol()	42
8.9.3.7	getTime()	42
8.9.3.8	getTimestamp()	42
8.9.3.9	getTypeName()	42
8.9.3.10	getTypeValue()	42
8.9.3.11	getURI()	43
8.9.3.12	setContent() [1/3]	43
8.9.3.13	setContent() [2/3]	43
8.9.3.14	setContent() [3/3]	43
8.9.3.15	setFrom()	43
8.10	yaodag::Version Class Reference	43
8.10.1	Detailed Description	44
8.10.2	Constructor & Destructor Documentation	44
8.10.2.1	Version() [1/3]	44
8.10.2.2	Version() [2/3]	44
8.10.2.3	Version() [3/3]	44
8.10.3	Member Function Documentation	44
8.10.3.1	getMajor()	44
8.10.3.2	getMinor()	44
8.10.3.3	getPatch()	45
8.10.3.4	getPreRelease()	45
8.10.3.5	getPreReleaseNumber()	45
8.11	yaodag::WebsocketClient Class Reference	45
8.11.1	Detailed Description	45

8.11.2 Constructor & Destructor Documentation	45
8.11.2.1 WebsocketClient()	46
8.11.2.2 ~WebsocketClient()	46
8.11.3 Member Function Documentation	46
8.11.3.1 logger()	46
8.11.3.2 loop()	46
8.11.3.3 start()	47
8.11.3.4 stop()	47
8.11.3.5 throwGeneralIfSameName()	47
8.12 yaodaq::WebsocketServer Class Reference	47
8.12.1 Detailed Description	48
8.12.2 Constructor & Destructor Documentation	48
8.12.2.1 WebsocketServer()	48
8.12.2.2 ~WebsocketServer()	48
8.12.3 Member Function Documentation	49
8.12.3.1 listen()	49
8.12.3.2 logger()	49
8.12.3.3 loop()	49
8.12.3.4 setVerbosity()	49
8.12.3.5 start()	49
8.12.3.6 stop()	50
<b>9 File Documentation</b>	<b>51</b>
9.1 docs/License.md File Reference	51
9.2 docs/Third-party licenses.md File Reference	51
9.3 yaodaq/Classification.hpp File Reference	51
9.4 Classification.hpp	51
9.5 yaodaq/ConnectionState.hpp File Reference	52
9.6 ConnectionState.hpp	52
9.7 yaodaq/Exception.hpp File Reference	53
9.8 Exception.hpp	53
9.9 yaodaq/Identifier.hpp File Reference	54
9.10 Identifier.hpp	54
9.11 yaodaq/Interrupt.hpp File Reference	54
9.12 Interrupt.hpp	55
9.13 yaodaq/IXWebsocketMessage.hpp File Reference	55
9.14 IXWebsocketMessage.hpp	55
9.15 yaodaq/Key.hpp File Reference	56
9.16 Key.hpp	56
9.17 yaodaq/LoggerHandler.hpp File Reference	56
9.18 LoggerHandler.hpp	57
9.19 yaodaq/Looper.hpp File Reference	57

9.20 Looper.hpp . . . . .	58
9.21 yaodaq/Message.hpp File Reference . . . . .	58
9.22 Message.hpp . . . . .	58
9.23 yaodaq/MessageType.hpp File Reference . . . . .	59
9.24 MessageType.hpp . . . . .	59
9.25 yaodaq/Severity.hpp File Reference . . . . .	60
9.26 Severity.hpp . . . . .	60
9.27 yaodaq/Signal.hpp File Reference . . . . .	60
9.28 Signal.hpp . . . . .	61
9.29 yaodaq/StatusCode.hpp File Reference . . . . .	61
9.30 StatusCode.hpp . . . . .	61
9.31 yaodaq/Version.hpp File Reference . . . . .	62
9.32 Version.hpp . . . . .	62
9.33 yaodaq/WebsocketClient.hpp File Reference . . . . .	62
9.34 WebsocketClient.hpp . . . . .	63
9.35 yaodaq/WebsocketServer.hpp File Reference . . . . .	63
9.36 WebsocketServer.hpp . . . . .	64
9.37 yaodaq/ConnectionState.cpp File Reference . . . . .	64
9.38 ConnectionState.cpp . . . . .	64
9.39 yaodaq/Exception.cpp File Reference . . . . .	65
9.40 Exception.cpp . . . . .	65
9.41 yaodaq/Identifier.cpp File Reference . . . . .	66
9.42 Identifier.cpp . . . . .	66
9.43 yaodaq/Interrupt.cpp File Reference . . . . .	67
9.44 Interrupt.cpp . . . . .	67
9.45 yaodaq/IXWebsocketMessage.cpp File Reference . . . . .	68
9.46 IXWebsocketMessage.cpp . . . . .	68
9.47 yaodaq/Key.cpp File Reference . . . . .	69
9.48 Key.cpp . . . . .	69
9.49 yaodaq/LoggerHandler.cpp File Reference . . . . .	69
9.50 LoggerHandler.cpp . . . . .	69
9.51 yaodaq/Looper.cpp File Reference . . . . .	70
9.52 Looper.cpp . . . . .	70
9.53 yaodaq/Message.cpp File Reference . . . . .	71
9.54 Message.cpp . . . . .	71
9.55 yaodaq/Version.cpp File Reference . . . . .	73
9.56 Version.cpp . . . . .	73
9.57 yaodaq/WebsocketClient.cpp File Reference . . . . .	73
9.58 WebsocketClient.cpp . . . . .	74
9.59 yaodaq/WebsocketServer.cpp File Reference . . . . .	75
9.60 WebsocketServer.cpp . . . . .	75



# Chapter 1

## License

Copyright (c) 2022 YAODAO

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 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](mailto: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](mailto:eay@cryptsoft.com)). This product includes software written by Tim
- Hudson ( [tjh@cryptsoft.com](mailto:tjh@cryptsoft.com)).
- \*/

### 2.1.2 Original SSLeay License

/\* Copyright (C) 1995-1998 Eric Young ( [eyay@cryptsoft.com](mailto:eyay@cryptsoft.com))

- All rights reserved.
- 
- This package is an SSL implementation written
- by Eric Young ( [eyay@cryptsoft.com](mailto:eyay@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](mailto: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 ([eyay@cryptsoft.com](mailto:eyay@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: nlomann. 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:

<a href="#">spdlog</a> . . . . .	19
<a href="#">yaodaq</a> . . . . .	19



## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

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

ix::ConnectionState	
yaodag::ConnectionState . . . . .	25
std::exception	
yaodag::Exception . . . . .	26
yaodag::Identifier . . . . .	28
yaodag::Interrupt . . . . .	30
yaodag::Key . . . . .	31
yaodag::LoggerHandler . . . . .	32
yaodag::Looper . . . . .	34
yaodag::Message . . . . .	36
yaodag::Open . . . . .	40
source_location	
yaodag::Exception . . . . .	26
semver::version	
yaodag::Version . . . . .	43
ix::WebSocket	
yaodag::WebsocketClient . . . . .	45
ix::WebSocketServer	
yaodag::WebsocketServer . . . . .	47



## Chapter 5

# Data Structure Index

### 5.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">yaodag::ConnectionState</a>	25
<a href="#">yaodag::Exception</a>	26
<a href="#">yaodag::Identifier</a>	28
<a href="#">yaodag::Interrupt</a>	30
<a href="#">yaodag::Key</a>	31
<a href="#">yaodag::LoggerHandler</a>	32
<a href="#">yaodag::Looper</a>	34
<a href="#">yaodag::Message</a>	36
<a href="#">yaodag::Open</a>	40
<a href="#">yaodag::Version</a>	43
<a href="#">yaodag::WebsocketClient</a>	45
<a href="#">yaodag::WebsocketServer</a>	47





## Chapter 6

# File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

yaodag/Classification.hpp	51
yaodag/ConnectionState.hpp	52
yaodag/Exception.hpp	53
yaodag/Identifier.hpp	54
yaodag/Interrupt.hpp	54
yaodag/IXWebsocketMessage.hpp	55
yaodag/Key.hpp	56
yaodag/LoggerHandler.hpp	56
yaodag/Looper.hpp	57
yaodag/Message.hpp	58
yaodag/MessageType.hpp	59
yaodag/Severity.hpp	60
yaodag/Signal.hpp	60
yaodag/StatusCode.hpp	61
yaodag/Version.hpp	62
yaodag/WebsocketClient.hpp	62
yaodag/WebsocketServer.hpp	63
yaodag/ConnectionState.cpp	64
yaodag/Exception.cpp	65
yaodag/Identifier.cpp	66
yaodag/Interrupt.cpp	67
yaodag/IXWebsocketMessage.cpp	68
yaodag/Key.cpp	69
yaodag/LoggerHandler.cpp	69
yaodag/Looper.cpp	70
yaodag/Message.cpp	71
yaodag/Version.cpp	73
yaodag/WebsocketClient.cpp	73
yaodag/WebsocketServer.cpp	75



## 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 [ConnectionState](#)
- class [Exception](#)
- class [Identifier](#)
- class [Interrupt](#)
- class [Key](#)
- class [LoggerHandler](#)
- class [Looper](#)
- class [Message](#)
- class [Open](#)
- 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 , [ConnectionError](#) = -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 yaodag::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 yaodag::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 yaodag::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 yaodag::MessageType : std::int_least16_t [strong]
```

**Enumerator**

Open	
Close	
ConnectionError	
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     ConnectionError = -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 yaodag::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 yaodag::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 yaodag::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 & yaodag::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 ) + 0; }
```





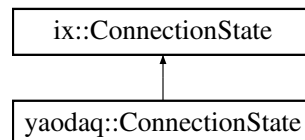
## Chapter 8

# Data Structure Documentation

### 8.1 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](#) ()

#### 8.1.1 Detailed Description

Definition at line 21 of file [ConnectionState.hpp](#).

#### 8.1.2 Constructor & Destructor Documentation

##### 8.1.2.1 ConnectionState()

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

Definition at line 14 of file [ConnectionState.cpp](#).

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

##### 8.1.2.2 ~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 }
```

#### 8.1.3 Member Function Documentation

### 8.1.3.1 computeId()

```
void yaodag::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(),
00031         m_Ids.end(), m_Pair );
00032         if( found == m_Ids.end() )
00033         {
00034             _id = id.getName();
00035             m_Ids.push_back( m_Pair );
00036         }
00037         else
00038         {
00039             setTerminated();
00040         }
00041     }
```

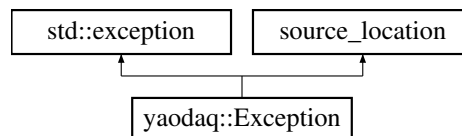
The documentation for this class was generated from the following files:

- [yaodag/ConnectionState.hpp](#)
- [yaodag/ConnectionState.cpp](#)

## 8.2 yaodag::Exception Class Reference

```
#include <yaodag/Exception.hpp>
```

Inheritance diagram for yaodag::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={})

### 8.2.1 Detailed Description

Definition at line 19 of file [Exception.hpp](#).

### 8.2.2 Constructor & Destructor Documentation

### 8.2.2.1 Exception() [1/2]

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

### 8.2.2.2 Exception() [2/2]

```
yaodag::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(); }
```

### 8.2.2.3 ~Exception()

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

## 8.2.3 Member Function Documentation

### 8.2.3.1 code()

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

Definition at line 20 of file [Exception.cpp](#).

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

### 8.2.3.2 description()

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

Definition at line 18 of file [Exception.cpp](#).

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

### 8.2.3.3 setFormat()

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

Definition at line 24 of file [Exception.hpp](#).

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

### 8.2.3.4 setStyle()

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

Definition at line 26 of file [Exception.hpp](#).

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

### 8.2.3.5 what()

```
const char * yaodag::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:

- [yaodag/Exception.hpp](#)
- [yaodag/Exception.cpp](#)

## 8.3 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

### Static Public Member Functions

- static [Identifier](#) [parse](#) (const std::string &)

#### 8.3.1 Detailed Description

Definition at line 16 of file [Identifier.hpp](#).

#### 8.3.2 Constructor & Destructor Documentation

##### 8.3.2.1 Identifier() [1/2]

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

##### 8.3.2.2 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 ) {}
```

#### 8.3.3 Member Function Documentation

##### 8.3.3.1 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 }
```

### 8.3.3.2 generateKey()

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

Definition at line 28 of file [Identifier.cpp](#).

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

### 8.3.3.3 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() ); }
```

### 8.3.3.4 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() ) ); }
```

### 8.3.3.5 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() ) ); }
```

### 8.3.3.6 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() ) ); }
```

### 8.3.3.7 getKey()

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

Definition at line 40 of file [Identifier.cpp](#).

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

### 8.3.3.8 getName()

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

Definition at line 38 of file [Identifier.cpp](#).

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

### 8.3.3.9 getType()

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

Definition at line 36 of file [Identifier.cpp](#).

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

### 8.3.3.10 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(),
00067                                magic_enum::enum_cast<Class>( result[1] ).value(), magic_enum::enum_cast<Family>( result[2] ).value()
00068         );
00069         return identifier;
00070     }
00071     else
00072     {
00073         throw Exception( StatusCode::WRONG_NUMBER_PARAMETERS, "Number of parameters in key should be 5
00074                           (Domain/Class/Family/Type/Name) !" );
00075     }
00076 }
```

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

- [yaodaq/Identifier.hpp](#)
- [yaodaq/Identifier.cpp](#)

## 8.4 yaodaq::Interrupt Class Reference

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

### Public Member Functions

- [Interrupt](#) ()
- void [init](#) ()
- void [restore](#) ()
- [Signal](#) [getSignal](#) ()
- [~Interrupt](#) ()

### 8.4.1 Detailed Description

Definition at line 19 of file [Interrupt.hpp](#).

### 8.4.2 Constructor & Destructor Documentation

#### 8.4.2.1 Interrupt()

```
yaodaq::Interrupt::Interrupt ( )
```

Definition at line 19 of file [Interrupt.cpp](#).

```
00019 { init(); }
```

### 8.4.2.2 ~Interrupt()

yaodaq::Interrupt::~~Interrupt ( )

Definition at line 42 of file [Interrupt.cpp](#).

```
00042 { restore(); }
```

## 8.4.3 Member Function Documentation

### 8.4.3.1 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 }
```

### 8.4.3.2 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 }
```

### 8.4.3.3 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](#)

## 8.5 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

### 8.5.1 Detailed Description

Definition at line 15 of file [Key.hpp](#).

### 8.5.2 Constructor & Destructor Documentation

#### 8.5.2.1 Key() [1/2]

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

#### 8.5.2.2 Key() [2/2]

```
yaodaq::Key::Key (
    const Domain & domain,
    const Class & c_class,
    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_class
    ) << 16 ) + static_cast<std::int_least16_t>( family ); }
```

### 8.5.3 Member Function Documentation

#### 8.5.3.1 getClass()

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

Definition at line 15 of file [Key.cpp](#).

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

#### 8.5.3.2 getDomain()

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

Definition at line 13 of file [Key.cpp](#).

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

#### 8.5.3.3 getFamily()

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

Definition at line 17 of file [Key.cpp](#).

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

#### 8.5.3.4 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](#)

## 8.6 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](#) ()

### 8.6.1 Detailed Description

Definition at line 21 of file [LoggerHandler.hpp](#).

### 8.6.2 Member Enumeration Documentation

#### 8.6.2.1 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 };
```

### 8.6.3 Constructor & Destructor Documentation

#### 8.6.3.1 LoggerHandler()

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

Definition at line 12 of file [LoggerHandler.cpp](#).

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

### 8.6.3.2 ~LoggerHandler()

yaodaq::LoggerHandler::~~LoggerHandler ( )  
 Definition at line 20 of file [LoggerHandler.cpp](#).  
 00020 {}

## 8.6.4 Member Function Documentation

### 8.6.4.1 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 }

### 8.6.4.2 clearSinks()

void yaodaq::LoggerHandler::clearSinks ( )  
 Definition at line 51 of file [LoggerHandler.cpp](#).  
 00052 {  
 00053     m\_Sinks.clear();  
 00054     init();  
 00055 }

### 8.6.4.3 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 ); }

### 8.6.4.4 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 }

### 8.6.4.5 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](#)

## 8.7 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](#) ()

### 8.7.1 Detailed Description

Definition at line 15 of file [Looper.hpp](#).

### 8.7.2 Constructor & Destructor Documentation

#### 8.7.2.1 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 }
```

#### 8.7.2.2 ~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 }
```

### 8.7.3 Member Function Documentation

#### 8.7.3.1 getInstance()

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

Definition at line 17 of file [Looper.cpp](#).

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

#### 8.7.3.2 getSignal()

[Signal](#) yaodaq::Looper::getSignal ( )

Definition at line 50 of file [Looper.cpp](#).

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

### 8.7.3.3 loop()

Signal yaodag::Looper::loop ( )

Definition at line 37 of file [Looper.cpp](#).

```
00038 {
00039     static Signal signal{ yaodag::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 == yaodag::Signal::NO );
00046     }
00047     return signal;
00048 }
```

### 8.7.3.4 supressInstance()

void yaodag::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 }
```

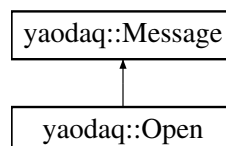
The documentation for this class was generated from the following files:

- [yaodag/Looper.hpp](#)
- [yaodag/Looper.cpp](#)

## 8.8 yaodag::Message Class Reference

#include <yaodag/Message.hpp>

Inheritance diagram for yaodag::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
- std::string [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)

### 8.8.1 Detailed Description

Definition at line 19 of file [Message.hpp](#).

### 8.8.2 Constructor & Destructor Documentation

#### 8.8.2.1 Message() [1/5]

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

Definition at line 23 of file [Message.cpp](#).

```
00024 {
00025     m_JSON["from"];
00026     m_JSON["to"];
00027     m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
00028     m_JSON["content"];
00029     m_JSON["timestamp"] = fmt::format( "{:%F %T %Z}", fmt::gmtime(
                                std::chrono::system_clock::to_time_t( std::chrono::system_clock::now() ) ) );
00030     m_JSON["meta"]["compiler"] = nlohmann::json::meta()["compiler"];
00031     m_JSON["meta"]["platform"] = nlohmann::json::meta()["platform"];
00032     m_JSON["meta"]["versions"]["json"] = nlohmann::json::meta()["version"]["string"];
00033     m_JSON["meta"]["versions"]["yaodaq"] = yaodaq_version.to_string();
00034     m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
00035 }
```

#### 8.8.2.2 Message() [2/5]

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

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

Definition at line 50 of file [Message.cpp](#).

```
00050 : Message( messageType ) { setContent( content ); }
```

#### 8.8.2.3 Message() [3/5]

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

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

Definition at line 52 of file [Message.cpp](#).

```
00052 : Message( messageType ) { setContent( content ); }
```

#### 8.8.2.4 Message() [4/5]

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

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

Definition at line 54 of file [Message.cpp](#).

```
00054 : Message( messageType ) { setContent( content ); }
```

### 8.8.2.5 Message() [5/5]

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

Definition at line 105 of file [Message.cpp](#).

```
00105 : Message() { m_JSON["type"] = magic_enum::enum_name( 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
```

Definition at line 56 of file [Message.cpp](#).

```
00056 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

### 8.8.3.2 get()

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

Definition at line 58 of file [Message.cpp](#).

```
00058 { return m_JSON; }
```

### 8.8.3.3 getContent()

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

Definition at line 64 of file [Message.cpp](#).

```
00065 {
00066     if( m_JSON["content"].is_null() ) return "";
00067     else if( m_JSON["content"].is_string() )
00068         return m_JSON["content"].get<std::string>();
00069     else
00070         return m_JSON["content"].dump();
00071 }
```

### 8.8.3.4 getIdentifier()

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

Definition at line 93 of file [Message.cpp](#).

```
00094 {
00095     if( m_JSON["from"].is_null() ) return {};
00096     else
00097     {
00098         Identifier id( m_JSON["from"]["type"].get<std::string>(),
m_JSON["from"]["name"].get<std::string>() );
00099         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(),
00100         magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
).value() );
00101         return id;
00102     }
00103 }
```

### 8.8.3.5 getTime()

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

Definition at line 75 of file [Message.cpp](#).

```
00076 {
00077     std::tm tm;
```

```

00078     memset( &tm, 0, sizeof( tm ) );
00079     std::stringstream ss( getTimestamp() );
00080     ss >> std::get_time( &tm, "%Y-%m-%d %H:%M:%S %Z" );
00081     return mktime( &tm );
00082 }

```

### 8.8.3.6 getTimestamp()

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

Definition at line 73 of file [Message.cpp](#).

```

00073 { return m_JSON["timestamp"].get<std::string>(); }

```

### 8.8.3.7 getTypeName()

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

Definition at line 60 of file [Message.cpp](#).

```

00060 { return m_JSON["type"].get<std::string>(); }

```

### 8.8.3.8 getTypeValue()

MessageType yaodaq::Message::getTypeValue ( ) const

Definition at line 62 of file [Message.cpp](#).

```

00062 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }

```

### 8.8.3.9 setContent() [1/3]

```

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

```

Definition at line 44 of file [Message.cpp](#).

```

00045 {
00046     m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00047     if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00048 }

```

### 8.8.3.10 setContent() [2/3]

```

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

```

Definition at line 37 of file [Message.cpp](#).

```

00037 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }

```

### 8.8.3.11 setContent() [3/3]

```

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

```

Definition at line 39 of file [Message.cpp](#).

```

00040 {
00041     m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00042     if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00043 }

```

### 8.8.3.12 setFrom()

```

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

```

Definition at line 84 of file [Message.cpp](#).

```

00085 {
00086     m_JSON["from"]["name"] = identifier.getName();
00087     m_JSON["from"]["type"] = identifier.getType();

```

```

00088     m_JSON["from"]["family"] = identifier.getFamily();
00089     m_JSON["from"]["class"]   = identifier.getClass();
00090     m_JSON["from"]["domain"]  = identifier.getDomain();
00091 }

```

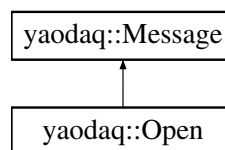
The documentation for this class was generated from the following files:

- [yaodaq/Message.hpp](#)
- [yaodaq/Message.cpp](#)

## 8.9 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
- std::string [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 [setContent](#) (const nlohmann::json &content)
- void [setContent](#) (const std::string &content)
- void [setContent](#) (const char \*content)

#### 8.9.1 Detailed Description

Definition at line 19 of file [IXWebsocketMessage.hpp](#).

#### 8.9.2 Constructor & Destructor Documentation



### 8.9.2.1 Open() [1/2]

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

Definition at line 10 of file [IXWebSocketMessage.cpp](#).

```
00010 : Message( MessageType::Open )
00011 {
00012     nlohmann::json j;
00013     j["uri"] = openInfo.uri;
00014     j["headers"] = openInfo.headers;
00015     j["protocol"] = openInfo.protocol;
00016     setContent( j );
00017 }
```

### 8.9.2.2 Open() [2/2]

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

Definition at line 19 of file [IXWebSocketMessage.cpp](#).

```
00019 :
00020     Open( openInfo )
00021 {
00022     nlohmann::json j = get();
00023     j["id"] = connectionState->getId();
00024     j["remote_ip"] = connectionState->getRemoteIp();
00025     j["remote_port"] = connectionState->getRemotePort();
00026     setContent( j );
00027 }
```

## 8.9.3 Member Function Documentation

### 8.9.3.1 dump()

```
std::string yaodag::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 56 of file [Message.cpp](#).

```
00056 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

### 8.9.3.2 get()

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

Definition at line 58 of file [Message.cpp](#).

```
00058 { return m_JSON; }
```

### 8.9.3.3 getContent()

```
std::string yaodag::Message::getContent ( ) const [inherited]
```

Definition at line 64 of file [Message.cpp](#).

```
00065 {
00066     if( m_JSON["content"].is_null() ) return "";
00067     else if( m_JSON["content"].is_string() )
00068         return m_JSON["content"].get<std::string>();
00069     else
00070         return m_JSON["content"].dump();
00071 }
```

#### 8.9.3.4 getHeaders()

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

Definition at line 30 of file [IXWebsocketMessage.cpp](#).

```
00031 {
00032     std::map<std::string, std::string> ret = get()["content"]["headers"].get<std::map<std::string,
std::string>()>;
00033     return ret;
00034 }
```

#### 8.9.3.5 getIdentifier()

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

Definition at line 93 of file [Message.cpp](#).

```
00094 {
00095     if( m_JSON["from"].is_null() ) return {};
00096     else
00097     {
00098         Identifier id( m_JSON["from"]["type"].get<std::string>(),
m_JSON["from"]["name"].get<std::string>() );
00099         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(),
00100             magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
).value() );
00101         return id;
00102     }
00103 }
```

#### 8.9.3.6 getProtocol()

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

Definition at line 36 of file [IXWebsocketMessage.cpp](#).

```
00036 { return get()["content"]["protocol"].get<std::string>(); }
```

#### 8.9.3.7 getTime()

std::time\_t yaodaq::Message::getTime ( ) const [inherited]

Definition at line 75 of file [Message.cpp](#).

```
00076 {
00077     std::tm tm;
00078     memset( &tm, 0, sizeof( tm ) );
00079     std::stringstream ss( getTimestamp() );
00080     ss >> std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00081     return mktime( &tm );
00082 }
```

#### 8.9.3.8 getTimestamp()

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

Definition at line 73 of file [Message.cpp](#).

```
00073 { return m_JSON["timestamp"].get<std::string>(); }
```

#### 8.9.3.9 getTypeName()

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

Definition at line 60 of file [Message.cpp](#).

```
00060 { return m_JSON["type"].get<std::string>(); }
```

#### 8.9.3.10 getTypeValue()

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

Definition at line 62 of file [Message.cpp](#).

```
00062 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.9.3.11 `getURI()`

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

Definition at line 28 of file [IXWebsocketMessage.cpp](#).

```
00028 { return get()["content"]["uri"].get<std::string>(); }
```

8.9.3.12 `setContent()` [1/3]

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

Definition at line 44 of file [Message.cpp](#).

```
00045 {
00046     m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00047     if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00048 }
```

8.9.3.13 `setContent()` [2/3]

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

Definition at line 37 of file [Message.cpp](#).

```
00037 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.9.3.14 `setContent()` [3/3]

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

Definition at line 39 of file [Message.cpp](#).

```
00040 {
00041     m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00042     if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00043 }
```

8.9.3.15 `setFrom()`

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

Definition at line 84 of file [Message.cpp](#).

```
00085 {
00086     m_JSON["from"]["name"] = identifier.getName();
00087     m_JSON["from"]["type"] = identifier.getType();
00088     m_JSON["from"]["family"] = identifier.getFamily();
00089     m_JSON["from"]["class"] = identifier.getClass();
00090     m_JSON["from"]["domain"] = identifier.getDomain();
00091 }
```

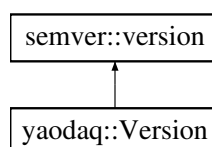
The documentation for this class was generated from the following files:

- [yaodag/IXWebsocketMessage.hpp](#)
- [yaodag/IXWebsocketMessage.cpp](#)

8.10 `yaodag::Version` Class Reference

```
#include <yaodag/Version.hpp>
```

Inheritance diagram for `yaodag::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](#) ()

### 8.10.1 Detailed Description

Definition at line 15 of file [Version.hpp](#).

### 8.10.2 Constructor & Destructor Documentation

#### 8.10.2.1 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 ) {}
```

#### 8.10.2.2 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 ) {}
```

#### 8.10.2.3 Version() [3/3]

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

### 8.10.3 Member Function Documentation

#### 8.10.3.1 getMajor()

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

Definition at line 12 of file [Version.cpp](#).

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

#### 8.10.3.2 getMinor()

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

Definition at line 14 of file [Version.cpp](#).

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

### 8.10.3.3 getPatch()

std::uint8\_t yaodaq::Version::getPatch ( )

Definition at line 16 of file [Version.cpp](#).

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

### 8.10.3.4 getPreRelease()

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

Definition at line 18 of file [Version.cpp](#).

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

### 8.10.3.5 getPreReleaseNumber()

std::uint8\_t yaodaq::Version::getPreReleaseNumber ( )

Definition at line 20 of file [Version.cpp](#).

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

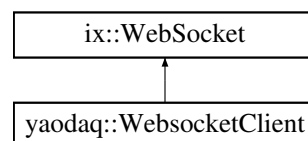
The documentation for this class was generated from the following files:

- [yaodaq/Version.hpp](#)
- [yaodaq/Version.cpp](#)

## 8.11 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](#) ()

### Static Public Member Functions

- static void [throwGeneralIfSameName](#) (const bool &)

### 8.11.1 Detailed Description

Definition at line 20 of file [WebsocketClient.hpp](#).

### 8.11.2 Constructor & Destructor Documentation

### 8.11.2.1 WebSocketClient()

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

Definition at line 24 of file [WebSocketClient.cpp](#).

```
00024         : m_Identifier( type,
00025             name )
00026 {
00027     ix::initNetSystem();
00028     m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
00029     m_Logger.setName( m_Identifier.get() );
00030     m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00031
00032     ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
00033     setExtraHeaders( header );
00034
00035     setOnMessageCallback(
00036         [this]( const ix::WebSocketMessagePtr& msg )
00037         {
00038             if( msg->type == ix::WebSocketMessageType::Message ) { logger()->error( "{}", msg->str ); }
00039             else if( msg->type == ix::WebSocketMessageType::Error )
00040             {
00041                 std::cout << "Connection error: " << msg->errorInfo.reason << std::endl;
00042             }
00043             else if( msg->type == ix::WebSocketMessageType::Close )
00044             {
00045                 disableAutomaticReconnection();
00046                 std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
00047                 if( msg->closeInfo.code == magic_enum::enum_integer(
00048                     StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ) )
00049                 {
00050                     logger()->critical( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold,
00051                         msg->closeInfo.reason ) );
00052                     if( m_ThrowGeneralIfSameName ) throw Exception(
00053                         StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED, msg->closeInfo.reason );
00054                 }
00055             }
00056         }
00057     );
00058 }
```

### 8.11.2.2 ~WebSocketClient()

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

Definition at line 58 of file [WebSocketClient.cpp](#).

```
00059 {
00060     stop();
00061     ix::uninitNetSystem();
00062 }
```

## 8.11.3 Member Function Documentation

### 8.11.3.1 logger()

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

Definition at line 29 of file [WebSocketClient.hpp](#).

```
00029 { return m_Logger.logger(); }
```

### 8.11.3.2 loop()

```
void yaodaq::WebSocketClient::loop ( )
```

Definition at line 83 of file [WebSocketClient.cpp](#).

```
00084 {
00085     WebSocketClient::start();
00086     m_Looper.supressInstance();
00087     onRaisingSignal();
00088 }
```

## 8.11.3.3 start()

void yaodaq::WebsocketClient::start ( )

Definition at line 64 of file [WebsocketClient.cpp](#).

```
00065 {
00066     if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00067     {
00068         logger()->trace( "Client started. Connected to {}", getUrl() );
00069         ix::WebSocket::start();
00070     }
00071 }
```

## 8.11.3.4 stop()

void yaodaq::WebsocketClient::stop ( )

Definition at line 73 of file [WebsocketClient.cpp](#).

```
00074 {
00075     if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00076     {
00077         logger()->trace( "Client stopped" );
00078         ix::WebSocket::stop();
00079         while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
                                std::chrono::microseconds( 1 ) ); }
00080     }
00081 }
```

## 8.11.3.5 throwGeneralIfSameName()

void yaodaq::WebsocketClient::throwGeneralIfSameName (   
 const bool & activate ) [static]

Definition at line 22 of file [WebsocketClient.cpp](#).

```
00022 { m_ThrowGeneralIfSameName = activate; }
```

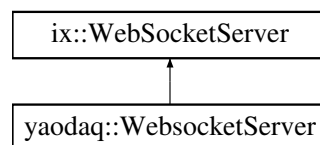
The documentation for this class was generated from the following files:

- [yaodaq/WebsocketClient.hpp](#)
- [yaodaq/WebsocketClient.cpp](#)

## 8.12 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 &handshakeTimeoutSecs=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](#) ( )
- void [setVerbosity](#) (const [yaodaq::LoggerHandler::Verbosity](#) &verbosity)
- std::shared\_ptr< spdlog::logger > [logger](#) ( )

## 8.12.1 Detailed Description

Definition at line 20 of file [WebsocketServer.hpp](#).

## 8.12.2 Constructor & Destructor Documentation

### 8.12.2.1 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 26 of file [WebsocketServer.cpp](#).

```
00026
:
00027 ix::WebsocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
    m_Identifier( type, name )
00028 {
00029     ix::initNetSystem();
00030
00031     m_Identifier.generateKey( Domain::Application, Class::Server, Family::WebsocketServer );
00032     m_Logger.setName( m_Identifier.get() );
00033     m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00034
00035     setConnectionFactory( []() { return std::make_shared<ConnectionState>(); } );
00036
00037     setOnClientMessageCallback(
00038         [this]( std::shared_ptr<ix::ConnectionState> connectionState, ix::Websocket& websocket, const
ix::WebsocketMessagePtr& msg )
00039         {
00040             // The ConnectionState object contains information about the connection
00041             std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>(
connectionState );
00042
00043             if( msg->type == ix::WebsocketMessageType::Open )
00044             {
00045                 // Check if a client with the same name is already connected;
00046                 logger()->critical( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold, getHost() + ":"
+ std::to_string( getPort() ) ) );
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                     return;
00054                 }
00055             }
00056             else if( msg->type == ix::WebsocketMessageType::Message )
00057             {
00058                 websocket.send( msg->str, msg->binary );
00059             }
00060         } );
00061 }
```

### 8.12.2.2 ~WebsocketServer()

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

Definition at line 101 of file [WebsocketServer.cpp](#).

```
00102 {
00103     stop();
```



```
00104     ix::uninitNetSystem();
00105 }
```

## 8.12.3 Member Function Documentation

### 8.12.3.1 listen()

```
void yaodq::WebsocketServer::listen ( )
```

Definition at line 63 of file [WebsocketServer.cpp](#).

```
00064 {
00065     if( !m_isListening )
00066     {
00067         std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
00068         if( ret.first )
00069         {
00070             m_isListening = ret.first;
00071             logger()->info( "Server listening on {0}:{1}", getHost(), getPort() );
00072         }
00073         else
00074             throw Exception( StatusCode::LISTEN_ERROR, ret.second );
00075     }
00076 }
```

### 8.12.3.2 logger()

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

Definition at line 34 of file [WebsocketServer.hpp](#).

```
00034 { return m_Logger.logger(); }
```

### 8.12.3.3 loop()

```
void yaodq::WebsocketServer::loop ( )
```

Definition at line 107 of file [WebsocketServer.cpp](#).

```
00108 {
00109     listen();
00110     start();
00111     m_Looper.supressInstance();
00112     onRaisingSignal();
00113 }
```

### 8.12.3.4 setVerbosity()

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

Definition at line 99 of file [WebsocketServer.cpp](#).

```
00099 { m_Logger.setVerbosity( verbosity ); }
```

### 8.12.3.5 start()

```
void yaodq::WebsocketServer::start ( )
```

Definition at line 78 of file [WebsocketServer.cpp](#).

```
00079 {
00080     if( !m_isStarted )
00081     {
00082         m_isStarted = true;
00083         logger()->trace( "Server started" );
00084         ix::WebSocketServer::start();
00085     }
00086 }
```

### 8.12.3.6 stop()

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

Definition at line 88 of file [WebsocketServer.cpp](#).

```
00089 {
00090     if( !m_isStopped )
00091     {
00092         m_isStopped = true;
00093         useless      = !useless;
00094         logger()->trace( "Server stopped" );
00095         ix::WebSocketServer::stop();
00096     }
00097 }
```

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 yaodag/Classification.hpp File Reference

```
#include <cstdint>
```

#### Namespaces

- namespace `yaodag`

#### Enumerations

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

### 9.4 Classification.hpp

[Go to the documentation of this file.](#)

```
00001 #ifndef YAODAG_CLASSIFICATION
00002 #define YAODAG_CLASSIFICATION
00003
00008 #include <cstdint>
00009
00010 namespace yaodag
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

```

## 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&
00029             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
00002 #define YAODAQ_IDENTIFIER
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_class =
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
00032 private:
00033     std::string m_Type{ "Unknown" };
00034     std::string m_Name{ "Unknown" };
00035     Key m_Key;
00036 };
00037
00038 } // namespace yaodaq
00039
00040 #endif // YAODAQ_IDENTIFIER
```

## 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
00002 #define YAODAQ_HANDLER
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

```

## 9.13 yaodaq/IXWebsocketMessage.hpp File Reference

```

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

```

## Data Structures

- class `yaodaq::Open`

## 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/IXWebSocketOpenInfo.h>
00012 #include <map>
00013 #include <memory>

```

```

00014 #include <string>
00015
00016 namespace yaodaq
00017 {
00018
00019 class Open : public Message
00020 {
00021 public:
00022     explicit Open( const ix::WebSocketOpenInfo& openInfo );
00023     Open( const ix::WebSocketOpenInfo& openInfo, std::shared_ptr<ConnectionState>& connectionState );
00024     std::string          getURI() const;
00025     std::map<std::string, std::string> getHeaders() const;
00026     std::string          getProtocol() const;
00027 };
00028
00029 } // namespace yaodaq
00030 #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
00002 #define YAODAQ_KEY
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_class, 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

```

## 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
00002 #ifndef YAODAQ_MESSAGE
00003 #define YAODAQ_MESSAGE
00004
00009 #include "nlohmann/json.hpp"
00010 #include "yaodaq/MessageType.hpp"
00011
00012 #include <string>
00013
00014 namespace yaodaq
00015 {
00016
00017 class Identifier;
00018
00019 class Message
00020 {
```

```

00021 public:
00022     Message();
00023     explicit Message( const nlohmann::json& content, const MessageType& messageType =
        MessageType::Unknown );
00024     explicit Message( const std::string& content, const MessageType& messageType = MessageType::Unknown
        );
00025     explicit Message( const char* content, const MessageType& messageType = MessageType::Unknown );
00026     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;
00027     nlohmann::json get() const;
00028     std::string getContent() const;
00029     std::string getTypeName() const;
00030     MessageType getTypeValue() const;
00031     std::string getTimestamp() const;
00032     std::time_t getTime() const;
00033     Identifier getIdentifier() const;
00034     void setFrom( const Identifier& );
00035
00036 protected:
00037     explicit Message( const MessageType& messageType );
00038     void setContent( const nlohmann::json& content );
00039     void setContent( const std::string& content );
00040     void setContent( const char* content );
00041
00042 private:
00043     nlohmann::json m_JSON;
00044 };
00045
00046 } // namespace yaodaq
00047
00048 #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::ConnectionError](#) = -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_MESSAGE_TYPE
00002 #define YAODAQ_MESSAGE_TYPE
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     ConnectionError = -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 ) + 0; }
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
```

## 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
00002 #define YAODAQ_VERSION
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

```

## 9.33 yaodaq/WebsocketClient.hpp File Reference

```

#include "yaodaq/Identifier.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/LoggerHandler.hpp"
00010 #include "yaodaq/Looper.hpp"
00011
00012 #include <ixwebsocket/IXWebSocket.h>
00013 #include <memory>
00014 #include <spdlog/spdlog.h>
00015 #include <string>
00016
00017 namespace yaodaq
00018 {
00019
00020 class WebsocketClient : public ix::WebSocket
00021 {
00022 public:
00023     static void throwGeneralIfSameName( const bool& );
00024     explicit WebsocketClient( const std::string& name, const std::string& type = "YAODAQWebsocketClient"
00025 );
00026     virtual ~WebsocketClient();
00027     void start();
00028     void stop();
00029     void loop();
00030     std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
00031 private:
00032     void onRaisingSignal();
00033     Identifier m_Identifier;
00034     LoggerHandler m_Logger;
00035     Looper m_Looper;
00036     static bool m_ThrowGeneralIfSameName;
00037 };
00038
00039 } // namespace yaodaq
00040
00041 #endif
```

## 9.35 yaodaq/WebsocketServer.hpp File Reference

```
#include "yaodaq/Identifier.hpp"
#include "yaodaq/LoggerHandler.hpp"
#include "yaodaq/Looper.hpp"
#include <ixwebsocket/IXWebSocketServer.h>
#include <memory>
#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 "yaodag/Identifier.hpp"
00009 #include "yaodag/LoggerHandler.hpp"
00010 #include "yaodag/Looper.hpp"
00011
00012 #include <ixwebsocket/IXWebSocketServer.h>
00013 #include <memory>
00014 #include <spdlog/spdlog.h>
00015 #include <string>
00016
00017 namespace yaodag
00018 {
00019
00020 class WebsocketServer : public ix::WebSocketServer
00021 {
00022 public:
00023     explicit WebsocketServer( const std::string& name, const int& port = ix::SocketServer::kDefaultPort,
00024                             const std::string& host = ix::SocketServer::kDefaultHost, const int& backlog =
00025                             ix::SocketServer::kDefaultTcpBacklog,
00026                             const std::size_t& maxConnections =
00027                             ix::SocketServer::kDefaultMaxConnections, const int& handshakeTimeoutSecs =
00028                             ix::WebSocketServer::kDefaultHandShakeTimeoutSecs, const int& addressFamily =
00029                             ix::SocketServer::kDefaultAddressFamily,
00030                             const std::string& type = "YAODAQWebsocketServer" );
00031     virtual ~WebsocketServer();
00032     void loop();
00033     void start();
00034     void stop( bool useless = true );
00035     void listen();
00036
00037     void setVerbosity( const yaodag::LoggerHandler::Verbosity& verbosity );
00038
00039     std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
00040
00041 private:
00042     void onRaisingSignal();
00043     bool m_isListening{ false };
00044     Identifier m_Identifier;
00045     LoggerHandler m_Logger;
00046     Interrupt m_Interrupt;
00047     Looper m_Looper;
00048     bool m_isStopped{ false };
00049     bool m_isStarted{ false };
00050 };
00051 } // namespace yaodag
00052
00053 #endif // YAODAQ_WEBSOCKETSERVER

```

## 9.37 yaodag/ConnectionState.cpp File Reference

```

#include "yaodag/ConnectionState.hpp"
#include "yaodag/Identifier.hpp"

```

### Namespaces

- namespace [yaodag](#)

## 9.38 ConnectionState.cpp

[Go to the documentation of this file.](#)

```

00001
00005 #include "yaodag/ConnectionState.hpp"
00006
00007 #include "yaodag/Identifier.hpp"
00008
00009 namespace yaodag
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_class, const Family& family ) {
    m_Key = Key( domain, c_class, 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 } // 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 ); } );
00059         break;
00060         case Signal::FPE: std::signal( SIGFPE, []( int ) -> void { m_Signal.store( Signal::FPE ); } );
00061         break;
00062         case Signal::ILL: std::signal( SIGILL, []( int ) -> void { m_Signal.store( Signal::ILL ); } );
00063         break;
00064         case Signal::SEGV: std::signal( SIGSEGV, []( int ) -> void { m_Signal.store( Signal::SEGV ); } );
00065         break;
00066         case Signal::INT: std::signal( SIGINT, []( int ) -> void { m_Signal.store( Signal::INT ); } );
00067         break;
00068         case Signal::TERM: std::signal( SIGTERM, []( int ) -> void { m_Signal.store( Signal::TERM ); } );
00069         break;
00070         default: break;
00071     }
00072 }
00073 // 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 Open::Open( const ix::WebSocketOpenInfo& openInfo ) : Message( MessageType::Open )
00011 {
00012     nlohmann::json j;
00013     j["uri"] = openInfo.uri;
00014     j["headers"] = openInfo.headers;
00015     j["protocol"] = openInfo.protocol;
00016     setContent( j );
00017 }
00018
00019 Open::Open( const ix::WebSocketOpenInfo& openInfo, std::shared_ptr<ConnectionState>& connectionState )
00020 : Open( openInfo )
00021 {
00022     nlohmann::json j = get();
00023     j["id"] = connectionState->getId();
00024     j["remote_ip"] = connectionState->getRemoteIp();
00025     j["remote_port"] = connectionState->getRemotePort();
00026     setContent( j );
00027 }
00028 std::string Open::getURI() const { return get()["content"]["uri"].get<std::string>(); }
00029
00030 std::map<std::string, std::string> Open::getHeaders() const
00031 {

```

```

00032     std::map<std::string, std::string> ret = get()["content"]["headers"].get<std::map<std::string,
std::string>>();
00033     return ret;
00034 }
00035
00036 std::string Open::getProtocol() const { return get()["content"]["protocol"].get<std::string>(); }
00037
00038 } // 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_class, const Family& family ) { m_Key = (
static_cast<std::int_least8_t>( domain ) << 24 ) + ( static_cast<std::int_least8_t>( c_class ) << 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 <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 <string>
00015
00016 // Versions numbers
00017 #include <ixwebsocket/IXWebSocketVersion.h>

```

```

00018 #include <yaodag/YaodagVersion.hpp>
00019
00020 namespace yaodag
00021 {
00022
00023 Message::Message()
00024 {
00025     m_JSON["from"];
00026     m_JSON["to"];
00027     m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
00028     m_JSON["content"];
00029     m_JSON["timestamp"] = fmt::format( "{:%F %T %z}", fmt::gmtime(
std::chrono::system_clock::to_time_t( std::chrono::system_clock::now() ) ) );
00030     m_JSON["meta"]["compiler"] = nlohmann::json::meta()["compiler"];
00031     m_JSON["meta"]["platform"] = nlohmann::json::meta()["platform"];
00032     m_JSON["meta"]["versions"]["json"] = nlohmann::json::meta()["version"]["string"];
00033     m_JSON["meta"]["versions"]["yaodag"] = yaodag_version.to_string();
00034     m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
00035 }
00036
00037 void Message::setContent( const nlohmann::json& content ) { m_JSON["content"] =
static_cast<nlohmann::json>( content ); }
00038
00039 void Message::setContent( const std::string& content )
00040 {
00041     m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00042     if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00043 }
00044 void Message::setContent( const char* content )
00045 {
00046     m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00047     if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00048 }
00049
00050 Message::Message( const nlohmann::json& content, const MessageType& messageType ) : Message(
messageType ) { setContent( content ); }
00051
00052 Message::Message( const std::string& content, const MessageType& messageType ) : Message( messageType
) { setContent( content ); }
00053
00054 Message::Message( const char* content, const MessageType& messageType ) : Message( messageType ) {
setContent( content ); }
00055
00056 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 ); }
00057
00058 nlohmann::json Message::get() const { return m_JSON; }
00059
00060 std::string Message::getTypeName() const { return m_JSON["type"].get<std::string>(); }
00061
00062 MessageType Message::getTypeValue() const { return magic_enum::enum_cast<MessageType>(
m_JSON["type"].get<std::string>() ).value(); }
00063
00064 std::string Message::getContent() const
00065 {
00066     if( m_JSON["content"].is_null() ) return "";
00067     else if( m_JSON["content"].is_string() )
00068         return m_JSON["content"].get<std::string>();
00069     else
00070         return m_JSON["content"].dump();
00071 }
00072
00073 std::string Message::getTimestamp() const { return m_JSON["timestamp"].get<std::string>(); }
00074
00075 std::time_t Message::getTime() const
00076 {
00077     std::tm tm;
00078     memset( &tm, 0, sizeof( tm ) );
00079     std::stringstream ss( getTimestamp() );
00080     ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00081     return mktime( &tm );
00082 }
00083
00084 void Message::setFrom( const Identifier& identifier )
00085 {
00086     m_JSON["from"]["name"] = identifier.getName();
00087     m_JSON["from"]["type"] = identifier.getType();
00088     m_JSON["from"]["family"] = identifier.getFamily();
00089     m_JSON["from"]["class"] = identifier.getClass();
00090     m_JSON["from"]["domain"] = identifier.getDomain();
00091 }
00092
00093 Identifier Message::getIdentifier() const
00094 {
00095     if( m_JSON["from"].is_null() ) return {};
00096     else

```



```

00097     {
00098         Identifier id( m_JSON["from"]["type"].get<std::string>(),
00099             m_JSON["from"]["name"].get<std::string>() );
00100         id.generateKey( magic_enum::enum_cast<Domain>( m_JSON["from"]["domain"].get<std::string>()
00101             ).value(), magic_enum::enum_cast<Class>( m_JSON["from"]["class"].get<std::string>() ).value(),
00102             magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
00103             ).value() );
00104         return id;
00105     }
00106 }
00107
00108 Message::Message( const MessageType& messageType ) : Message() { m_JSON["type"] =
00109     magic_enum::enum_name( messageType ); }
00110
00111 // 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 bool WebSocketClient::m_ThrowGeneralIfSameName{ true };
00021
00022 void WebSocketClient::throwGeneralIfSameName( const bool& activate ) { m_ThrowGeneralIfSameName =
    activate; }
00023
00024 WebSocketClient::WebSocketClient( const std::string& name, const std::string& type ) : m_Identifier(
    type, name )
00025 {
00026     ix::initNetSystem();
00027
00028     m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
00029     m_Logger.setName( m_Identifier.get() );
00030     m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00031
00032     ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
00033     setExtraHeaders( header );
00034
00035     setOnMessageCallback(
00036         [this]( const ix::WebSocketMessagePtr& msg )
00037         {
00038             if( msg->type == ix::WebSocketMessageType::Message ) { logger()->error( "{}", msg->str ); }
00039             else if( msg->type == ix::WebSocketMessageType::Error )
00040             {
00041                 std::cout << "Connection error: " << msg->errorInfo.reason << std::endl;
00042             }
00043             else if( msg->type == ix::WebSocketMessageType::Close )
00044             {
00045                 disableAutomaticReconnection();
00046                 std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
00047                 if( msg->closeInfo.code == magic_enum::enum_integer(
    StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ) )
00048                 {
00049                     logger()->critical( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold,
    msg->closeInfo.reason ) );
00050                     if( m_ThrowGeneralIfSameName ) throw Exception(
    StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED, msg->closeInfo.reason );
00051                 }
00052             }
00053         } );
00054
00055     };
00056 }
00057
00058 WebSocketClient::~WebSocketClient()
00059 {
00060     stop();
00061     ix::uninitNetSystem();
00062 }
00063
00064 void WebSocketClient::start()
00065 {
00066     if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00067     {
00068         logger()->trace( "Client started. Connected to {}", getUrl() );
00069         ix::WebSocket::start();
00070     }
00071 }
00072
00073 void WebSocketClient::stop()
00074 {
00075     if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00076     {
00077         logger()->trace( "Client stopped" );
00078         ix::WebSocket::stop();
00079         while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
    std::chrono::microseconds( 1 ) ); }
00080     }

```

```

00081 }
00082
00083 void WebsocketClient::loop()
00084 {
00085     WebsocketClient::start();
00086     m_Looper.supressInstance();
00087     onRaisingSignal();
00088 }
00089
00090 void WebsocketClient::onRaisingSignal()
00091 {
00092     Signal signal = m_Looper.loop();
00093     if( m_Looper.getInstance() == 0 )
00094     {
00095         int value = magic_enum::enum_integer( signal );
00096         if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
00097 "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00098         else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
00099         {
00100             logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00101         }
00102         else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
00103         {
00104             fmt::print( "\n" );
00105             logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00106         }
00107         else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
00108         {
00109             fmt::print( "\n" );
00110             logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00111         }
00112         else
00113         {
00114             fmt::print( "\n" );
00115             logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00116         }
00117         if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
00118             std::exit( magic_enum::enum_integer( signal ) );
00119     }
00120 } // namespace yaodaq

```

## 9.59 yaodaq/WebsocketServer.cpp File Reference

```

#include "yaodaq/WebsocketServer.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/ConnectionState.hpp"
00008 #include "yaodaq/Exception.hpp"
00009 #include "yaodaq/IXWebsocketMessage.hpp"
00010 #include "yaodaq/Identifier.hpp"
00011 #include "yaodaq/StatusCode.hpp"
00012
00013 #include <chrono>
00014 #include <iostream>
00015 #include <ixwebsocket/IXNetSystem.h>
00016 #include <magic_enum.hpp>
00017 #include <spdlog/sinks/stdout_color_sinks.h>
00018 #include <spdlog/spdlog.h>
00019 #include <string>
00020 #include <thread>
00021 #include <utility>
00022
00023 namespace yaodaq
00024 {
00025
00026 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 ) :
00027     ix::WebsocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
    m_Identifier( type, name )
00028 {
00029     ix::initNetSystem();
00030
00031     m_Identifier.generateKey( Domain::Application, Class::Server, Family::WebsocketServer );
00032     m_Logger.setName( m_Identifier.get() );
00033     m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00034
00035     setConnectionFactory( []() { return std::make_shared<ConnectionState>(); } );
00036
00037     setOnClientMessageCallback(
00038         [this]( std::shared_ptr<ix::ConnectionState> connectionState, ix::Websocket& websocket, const
    ix::WebsocketMessagePtr& msg )
00039         {
00040             // The ConnectionState object contains information about the connection
00041             std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>(
    connectionState );
00042
00043             if( msg->type == ix::WebsocketMessageType::Open )
00044             {
00045                 // Check if a client with the same name is already connected;
00046                 logger()->critical( fmt::format( fg( fmt::color::red ) | fmt::emphasis::bold, getHost() + ":"
    + std::to_string( getPort() ) ) );
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                     return;
00054                 }
00055             }
00056             else if( msg->type == ix::WebsocketMessageType::Message )
00057             {
00058                 websocket.send( msg->str, msg->binary );
00059             }
00060             } );
00061     }
00062
00063 void WebsocketServer::listen()
00064 {
00065     if( !m_isListening )
00066     {
00067         std::pair<bool, std::string> ret = ix::WebsocketServer::listen();
00068         if( ret.first )
00069         {
00070             m_isListening = ret.first;
00071             logger()->info( "Server listening on {}:{1}", getHost(), getPort() );
00072         }
00073         else
00074             throw Exception( StatusCode::LISTEN_ERROR, ret.second );
00075     }
00076 }
00077
00078 void WebsocketServer::start()
00079 {
00080     if( !m_isStarted )

```

```

00081     {
00082         m_isStarted = true;
00083         logger()->trace( "Server started" );
00084         ix::WebSocketServer::start();
00085     }
00086 }
00087
00088 void WebsocketServer::stop( bool useless )
00089 {
00090     if( !m_isStopped )
00091     {
00092         m_isStopped = true;
00093         useless      = !useless;
00094         logger()->trace( "Server stopped" );
00095         ix::WebSocketServer::stop();
00096     }
00097 }
00098
00099 void WebsocketServer::setVerbosity( const yaodag::LoggerHandler::Verbosity& verbosity ) {
00100     m_Logger.setVerbosity( verbosity ); }
00101
00102 WebsocketServer::~WebsocketServer()
00103 {
00104     stop();
00105     ix::uninitNetSystem();
00106 }
00107
00108 void WebsocketServer::loop()
00109 {
00110     listen();
00111     start();
00112     m_Looper.supressInstance();
00113     onRaisingSignal();
00114 }
00115
00116 void WebsocketServer::onRaisingSignal()
00117 {
00118     Signal signal = m_Looper.loop();
00119     if( m_Looper.getInstance() == 0 )
00120     {
00121         int value = magic_enum::enum_integer( signal );
00122         if( value >= magic_enum::enum_integer( yaodag::Severity::Critical ) ) { logger()->critical(
00123             "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00124         else if( value >= magic_enum::enum_integer( yaodag::Severity::Error ) )
00125         {
00126             logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00127         }
00128         else if( value >= magic_enum::enum_integer( yaodag::Severity::Warning ) )
00129         {
00130             fmt::print( "\n" );
00131             logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00132         }
00133         else if( value >= magic_enum::enum_integer( yaodag::Severity::Info ) )
00134         {
00135             fmt::print( "\n" );
00136             logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00137         }
00138         else
00139         {
00140             fmt::print( "\n" );
00141             logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00142             if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
00143                 std::exit( magic_enum::enum_integer( signal ) );
00144         }
00145     }
00146 }
00147 // namespace yaodag

```

