YAODAQ

Yet An Other DAQ

Generated by Doxygen 1.9.3

Chapter 1

License

Chapter 2

Third-party licenses

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

MIT License

Copyright (c) 2021 Lars Melchior

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

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

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

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

MIT License

Copyright (c) 2019 - 2021 Daniil Goncharov

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

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

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

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

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

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

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

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

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

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

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

2.1 LICENSE ISSUES

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

2.1.1 OpenSSL License

/* ==

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

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

2.1.2 Original SSLeay License

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 spdlog Namespace Reference

Typedefs

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

7.1.1 Detailed Description

Copyright

Copyright 2022 flagarde

7.1.2 Typedef Documentation

7.1.2.1 sink_ptr

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

Definition at line 15 of file LoggerHandler.hpp.

7.2 yaodaq Namespace Reference

Data Structures

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

Enumerations

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

Functions

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

7.2.1 Detailed Description

Copyright

Copyright 2022 flagarde

7.2.2 Enumeration Type Documentation

7.2.2.1 Class

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

Enumerator

Unknown	
Server	
Client	
Module	
Board	

Definition at line 22 of file Classification.hpp.

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

7.2.2.2 Domain

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

Enumerator

Unknown	
Application	
Web	

Definition at line 14 of file Classification.hpp.

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

7.2.2.3 Family

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

Enumerator

Unknown	
WebSocketServer	
WebSocketClient	
Logger	
Controller	
Configurator	
SlowController	
Viewer	
Analyser	
FileWriter	

Definition at line 34 of file Classification.hpp.

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

7.2.2.4 MessageType

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

Enumerator

Open	
Close	
Error	
Ping	
Pong	
Fragment	

Enumerator

Unknown	
Exception	

Definition at line 15 of file MessageType.hpp.

```
00016 {
00017   // IXWebSocket MessageType (Message is not set here)
00018   Open = -6,
00019   Close,
00020   Error,
00021   Ping,
00022   Pong,
00023   Fragment,
00024   // Unknown should not be used !
00025   Unknown = 0,
00026   Exception,
00027 };
```

7.2.2.5 Severity

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

Enumerator

Info	
Warning	
Error	
Critical	

Definition at line 13 of file Severity.hpp.

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

7.2.2.6 Signal

`enum class yaodaq::Signal [strong]`

Enumerator

NO	
ABRT	
FPE	
ILL	
SEGV	
INT	
TERM	

Definition at line 15 of file Signal.hpp.

```
00016 {
00017   NO   = 0,  // No Signal.
00018   // Critical
00019   ABRT = static_cast<int>( Severity::Critical ) + 1,  // (Signal Abort) Abnormal termination, such as
        is initiated by the abort function.
00020   FPE  = static_cast<int>( Severity::Critical ) + 2,  // (Signal Floating-Point Exception) Erroneous
        arithmetic operation, such as zero divide or an operation resulting in overflow (not necessarily with
        a floating-point operation).
00021   ILL  = static_cast<int>( Severity::Critical ) + 3,  // (Signal Illegal Instruction) Invalid function
        image, such as an illegal instruction. This is generally due to a corruption in the code or to an
        attempt to execute data.
00022   SEGV = static_cast<int>( Severity::Critical ) + 4,  // (Signal Segmentation Violation) Invalid
```

```
      access to storage: When a program tries to read or write outside the memory it has allocated.
00023   // Warning
00024   INT  = static_cast<int>( Severity::Warning ) + 1,  // (Signal Interrupt) Interactive attention
      signal. Generally generated by the application user.
00025   TERM = static_cast<int>( Severity::Warning ) + 2,  // (Signal Terminate) Termination request sent to
      program.
00026 };
```

7.2.2.7 StatusCode

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

Enumerator

SUCCESS	
LISTEN_ERROR	
WRONG_NUMBER_PARAMETERS	
CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED	

Definition at line 13 of file StatusCode.hpp.

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

7.2.3 Function Documentation

7.2.3.1 operator<<()

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

Definition at line 29 of file MessageType.hpp.

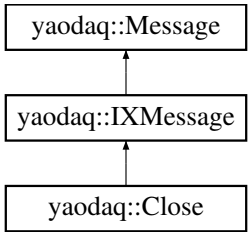
```
00029 { return os « static_cast<std::int_least8_t>( messageTypes ); }
```

Chapter 8

Data Structure Documentation

8.1 yaodaq::Close Class Reference

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



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.1.1 Detailed Description

Definition at line 44 of file IXWebsocketMessage.hpp.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 Close() [1/2]

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

Definition at line 60 of file IXWebsocketMessage.cpp.

```
00060                                                   : IXMessage( MessageType::Close )
00061 {
00062   nlohmann::json j;
00063   j["code"]   = closeInfo.code;
00064   j["reason"] = closeInfo.reason;
00065   j["remote"] = closeInfo.remote;
00066   setContent( j );
00067 }
```

8.1.2.2 Close() [2/2]

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

Definition at line 69 of file IXWebsocketMessage.cpp.

```
00069 : Close( closeInfo ) { setConnectionStateInfos( connectionState ); }
```

8.1.3 Member Function Documentation

8.1.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.1.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.1.3.3 getCode()

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

Definition at line 71 of file IXWebsocketMessage.cpp.

```
00071 { return get()["content"]["code"].get<std::uint16_t>(); }
```

8.1.3.4 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.1.3.5 getId()

```
std::string yaodaq::IXMessage::getId ( ) const  [inherited]
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 { return get()["content"]["id"].get<std::string>(); }
```

8.1.3.6 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.1.3.7 getReason()

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

Definition at line 72 of file IXWebsocketMessage.cpp.

```
00072 { return get()["content"]["reason"].get<std::string>(); }
```

8.1.3.8 getRemote()

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

Definition at line 73 of file IXWebsocketMessage.cpp.

```
00073 { return get()["content"]["remote"].get<bool>(); }
```

8.1.3.9 getRemoteIp()

```
std::string yaodaq::IXMessage::getRemoteIp ( ) const  [inherited]
```

Definition at line 33 of file IXWebsocketMessage.cpp.

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

8.1.3.10 getRemotePort()

```
int yaodaq::IXMessage::getRemotePort ( ) const  [inherited]
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00035 { return get()["content"]["remote_port"].get<int>(); }
```

8.1.3.11 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
```

```
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.1.3.12 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.1.3.13 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.1.3.14 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.1.3.15 setConnectionStateInfos()

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

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
```

8.1.3.16 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.1.3.17 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.1.3.18 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.1.3.19 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.1.4 Field Documentation

8.1.4.1 m_JSON

```
nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]
```

Definition at line 41 of file Message.hpp.

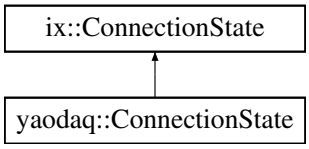
The documentation for this class was generated from the following files:

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

8.2 yaodaq::ConnectionState Class Reference

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

Inheritance diagram for yaodaq::ConnectionState:



Public Member Functions

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

8.2.1 Detailed Description

Definition at line 21 of file ConnectionState.hpp.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 ConnectionState()

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

Definition at line 14 of file ConnectionState.cpp.

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

8.2.2.2 ~ConnectionState()

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

Definition at line 16 of file ConnectionState.cpp.

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

8.2.3 Member Function Documentation

8.2.3.1 computeId()

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

Definition at line 22 of file ConnectionState.cpp.

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

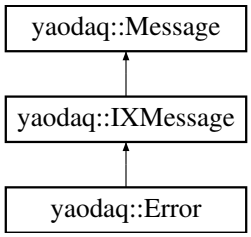
The documentation for this class was generated from the following files:

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

8.3 yaodaq::Error Class Reference

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

Inheritance diagram for yaodaq::Error:



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.3.1 Detailed Description

Definition at line 54 of file IXWebsocketMessage.hpp.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 Error() [1/2]

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

Definition at line 76 of file IXWebsocketMessage.cpp.

```
00076                                                 : IXMessage( MessageType::Error )
00077 {
00078   nlohmann::json j;
00079   j["retries"]             = errorInfo.retries;
00080   j["wait_time"]           = errorInfo.wait_time;
00081   j["http_status"]         = errorInfo.http_status;
00082   j["reason"]              = errorInfo.reason;
00083   j["decompression_error"] = errorInfo.decompressionError;
00084   setContent( j );
00085 }
```

8.3.2.2 Error() [2/2]

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

Definition at line 87 of file IXWebsocketMessage.cpp.

```
00087 : Error( errorInfo ) { setConnectionStateInfos( connectionState ); }
```

8.3.3 Member Function Documentation

8.3.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.3.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.3.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.3.3.4 getDecompressionError()

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

Definition at line 97 of file IXWebsocketMessage.cpp.

```
00097 { return get()["content"]["decompression_error"].get<bool>(); }
```

8.3.3.5 getHttpStatus()

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

Definition at line 93 of file IXWebsocketMessage.cpp.

```
00093 { return get()["content"]["http_status"].get<int>(); }
```

8.3.3.6 getId()

```
std::string yaodaq::IXMessage::getId ( ) const  [inherited]
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 { return get()["content"]["id"].get<std::string>(); }
```

8.3.3.7 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
      m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
      ).value() );
00098     return id;
00099   }
00100 }
```

8.3.3.8 getReason()

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

Definition at line 95 of file IXWebsocketMessage.cpp.

```
00095 { return get()["content"]["reason"].get<std::string>(); }
```

8.3.3.9 getRemoteIp()

```
std::string yaodaq::IXMessage::getRemoteIp ( ) const  [inherited]
```

Definition at line 33 of file IXWebsocketMessage.cpp.

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

8.3.3.10 getRemotePort()

```
int yaodaq::IXMessage::getRemotePort ( ) const  [inherited]
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00035 { return get()["content"]["remote_port"].get<int>(); }
```

8.3.3.11 getRetries()

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

Definition at line 89 of file IXWebsocketMessage.cpp.

```
00089 { return get()["content"]["retries"].get<std::uint16_t>(); }
```

8.3.3.12 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.3.3.13 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.3.3.14 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.3.3.15 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.3.3.16 getWaitTime()

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

Definition at line 91 of file IXWebsocketMessage.cpp.

```
00091 { return get()["content"]["wait_time"].get<double>(); }
```

8.3.3.17 setConnectionStateInfos()

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

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
```

8.3.3.18 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.3.3.19 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.3.3.20 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.3.3.21 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.3.4 Field Documentation

8.3.4.1 m_JSON

```
nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]
```

Definition at line 41 of file Message.hpp.

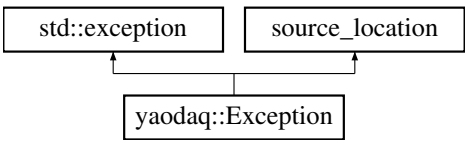
The documentation for this class was generated from the following files:

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

8.4 yaodaq::Exception Class Reference

`#include <yaodaq/Exception.hpp>`

Inheritance diagram for yaodaq::Exception:



Public Member Functions

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

Static Public Member Functions

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

8.4.1 Detailed Description

Definition at line 19 of file Exception.hpp.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 Exception() [1/2]

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

8.4.2.2 Exception() [2/2]

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

Definition at line 14 of file Exception.cpp.

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

8.4.2.3 ~Exception()

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

8.4.3 Member Function Documentation

8.4.3.1 code()

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

Definition at line 20 of file Exception.cpp.

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

8.4.3.2 description()

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

Definition at line 18 of file Exception.cpp.

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

8.4.3.3 setFormat()

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

Definition at line 24 of file Exception.hpp.

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

8.4.3.4 setStyle()

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

Definition at line 26 of file Exception.hpp.

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

8.4.3.5 what()

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

Definition at line 16 of file Exception.cpp.

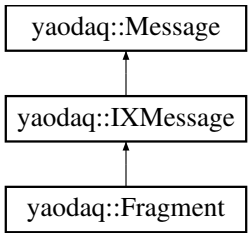
```
00016 { return m_Message.c_str(); }
```

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

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

8.5 yaodaq::Fragment Class Reference

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



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.5.1 Detailed Description

Definition at line 80 of file IXWebsocketMessage.hpp.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Fragment() [1/2]

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

Definition at line 110 of file IXWebsocketMessage.cpp.

```
00110 : IXMessage( MessageType::Fragment ) {}
```

8.5.2.2 Fragment() [2/2]

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

Definition at line 112 of file IXWebsocketMessage.cpp.

```
00112 : Fragment( fragment ) { setConnectionStateInfos( connectionState ); }
```

8.5.3 Member Function Documentation

8.5.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.5.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.5.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.5.3.4 getId()

```
std::string yaodaq::IXMessage::getId ( ) const  [inherited]
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 { return get()["content"]["id"].get<std::string>(); }
```

8.5.3.5 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.5.3.6 getRemoteIp()

```
std::string yaodaq::IXMessage::getRemoteIp ( ) const  [inherited]
```

Definition at line 33 of file IXWebsocketMessage.cpp.

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

8.5.3.7 getRemotePort()

```
int yaodaq::IXMessage::getRemotePort ( ) const  [inherited]
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00035 { return get()["content"]["remote_port"].get<int>(); }
```

8.5.3.8 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.5.3.9 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.5.3.10 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.5.3.11 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.5.3.12 setConnectionStateInfos()

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

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
```

8.5.3.13 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.5.3.14 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.5.3.15 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.5.3.16 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.5.4 Field Documentation

8.5.4.1 m_JSON

```
nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]
```

Definition at line 41 of file Message.hpp.

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

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

8.6 yaodaq::Identifier Class Reference

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

Public Member Functions

- Identifier ()=default
- Identifier (const std::string &type, const std::string &name)
- void generateKey (const Domain &domain=Domain::Unknown, const Class &c_lass=Class::Unknown, const Family &family=Family::Unknown)
- std::string getDomain () const
- std::string getClass () const

- std::string getFamily () const
- std::string getType () const
- std::string getName () const
- Key getKey () const
- std::string get () const
- bool empty () const
- bool operator< (const Identifier &) const

Static Public Member Functions

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

8.6.1 Detailed Description

Definition at line 16 of file Identifier.hpp.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 Identifier() [1/2]

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

8.6.2.2 Identifier() [2/2]

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

Definition at line 26 of file Identifier.cpp.

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

8.6.3 Member Function Documentation

8.6.3.1 empty()

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

Definition at line 19 of file Identifier.cpp.

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

8.6.3.2 generateKey()

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

Definition at line 28 of file Identifier.cpp.

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

8.6.3.3 get()

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

Definition at line 42 of file Identifier.cpp.

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

8.6.3.4 getClass()

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

Definition at line 32 of file Identifier.cpp.

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

8.6.3.5 getDomain()

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

Definition at line 30 of file Identifier.cpp.

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

8.6.3.6 getFamily()

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

Definition at line 34 of file Identifier.cpp.

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

8.6.3.7 getKey()

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

Definition at line 40 of file Identifier.cpp.

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

8.6.3.8 getName()

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

Definition at line 38 of file Identifier.cpp.

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

8.6.3.9 getType()

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

Definition at line 36 of file Identifier.cpp.

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

8.6.3.10 operator<()

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

Definition at line 75 of file Identifier.cpp.

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

8.6.3.11 parse()

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

Definition at line 44 of file Identifier.cpp.

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

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

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

8.7 yaodaq::Interrupt Class Reference

`#include <yaodaq/Interrupt.hpp>`

Public Member Functions

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

8.7.1 Detailed Description

Definition at line 19 of file Interrupt.hpp.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 Interrupt()

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

Definition at line 19 of file Interrupt.cpp.

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

8.7.2.2 ~Interrupt()

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

Definition at line 42 of file Interrupt.cpp.

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

8.7.3 Member Function Documentation

8.7.3.1 getSignal()

```
Signal yaodaq::Interrupt::getSignal ( )
```

Definition at line 44 of file Interrupt.cpp.

```
00045 {
00046   if( m_Signal.load() != Signal::NO )
00047   {
00048     std::lock_guard<std::mutex> guard( m_mutex );
00049     init();
00050   }
00051   return m_Signal.load();
00052 }
```

8.7.3.2 init()

```
void yaodaq::Interrupt::init ( )
```

Definition at line 31 of file Interrupt.cpp.

```
00032 {
00033   setSignal( Signal::TERM );
00034   setSignal( Signal::TERM );
00035   setSignal( Signal::SEGV );
00036   setSignal( Signal::INT );
00037   setSignal( Signal::ILL );
00038   setSignal( Signal::ABRT );
00039   setSignal( Signal::FPE );
00040 }
```

8.7.3.3 restore()

```
void yaodaq::Interrupt::restore ( )
```

Definition at line 21 of file Interrupt.cpp.

```
00022 {
00023   std::signal( SIGTERM, SIG_DFL );
00024   std::signal( SIGSEGV, SIG_DFL );
00025   std::signal( SIGINT, SIG_DFL );
00026   std::signal( SIGILL, SIG_DFL );
00027   std::signal( SIGABRT, SIG_DFL );
00028   std::signal( SIGFPE, SIG_DFL );
00029 }
```

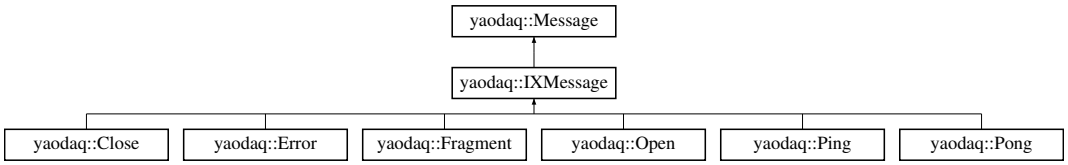
The documentation for this class was generated from the following files:

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

8.8 yaodaq::IXMessage Class Reference

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

Inheritance diagram for yaodaq::IXMessage:



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.8.1 Detailed Description

Definition at line 22 of file IXWebsocketMessage.hpp.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 IXMessage() [1/2]

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

Definition at line 10 of file IXWebsocketMessage.cpp.

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

8.8.2.2 IXMessage() [2/2]

```
yaodaq::IXMessage::IXMessage (
            const ix::WebSocketMessagePtr & msg )  [explicit]
```

Definition at line 12 of file IXWebsocketMessage.cpp.

```
00012                                                 : Message()
00013 {
00014   // FIXME
00015   nlohmann::json json = nlohmann::json::parse( msg->str, nullptr, false );
00016   if( json.is_discarded() ) { m_JSON["content"] = static_cast<std::string>( msg->str ); }
00017   else
00018     m_JSON = json;
00019   std::cout << m_JSON.dump() << std::endl;
00020 }
```

8.8.3 Member Function Documentation

8.8.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.8.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.8.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.8.3.4 getId()

```
std::string yaodaq::IXMessage::getId ( ) const
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 { return get()["content"]["id"].get<std::string>(); }
```

8.8.3.5 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.8.3.6 getRemoteIp()

```
std::string yaodaq::IXMessage::getRemoteIp ( ) const
```

Definition at line 33 of file IXWebsocketMessage.cpp.

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

8.8.3.7 getRemotePort()

```
int yaodaq::IXMessage::getRemotePort ( ) const
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00035 { return get()["content"]["remote_port"].get<int>(); }
```

8.8.3.8 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.8.3.9 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.8.3.10 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.8.3.11 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.8.3.12 setConnectionStateInfos()

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

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
```

8.8.3.13 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.8.3.14 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.8.3.15 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.8.3.16 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.8.4 Field Documentation

8.8.4.1 m_JSON

```
nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]
```

Definition at line 41 of file Message.hpp.

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

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

8.9 yaodaq::Key Class Reference

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

Public Member Functions

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

8.9.1 Detailed Description

Definition at line 15 of file Key.hpp.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 Key() [1/2]

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

8.9.2.2 Key() [2/2]

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

Definition at line 11 of file Key.cpp.

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

8.9.3 Member Function Documentation

8.9.3.1 getClass()

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

Definition at line 15 of file Key.cpp.

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

8.9.3.2 getDomain()

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

Definition at line 13 of file Key.cpp.

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

8.9.3.3 getFamily()

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

Definition at line 17 of file Key.cpp.

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

8.9.3.4 getKey()

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

Definition at line 19 of file Key.cpp.

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

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

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

8.10 yaodaq::LoggerHandler Class Reference

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

Public Types

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

Public Member Functions

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

8.10.1 Detailed Description

Definition at line 21 of file LoggerHandler.hpp.

8.10.2 Member Enumeration Documentation

8.10.2.1 Verbosity

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

Enumerator

Off	
Trace	
Debug	
Info	
Warn	
Error	
Critical	

Definition at line 24 of file LoggerHandler.hpp.

```
00025   {
00026     Off,
00027     Trace,
00028     Debug,
00029     Info,
00030     Warn,
00031     Error,
00032     Critical
00033   };
```

8.10.3 Constructor & Destructor Documentation

8.10.3.1 LoggerHandler()

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

Definition at line 12 of file LoggerHandler.cpp.

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

8.10.3.2 ~LoggerHandler()

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

Definition at line 20 of file LoggerHandler.cpp.

```
00020 {}
```

8.10.4 Member Function Documentation

8.10.4.1 addSink()

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

Definition at line 45 of file LoggerHandler.cpp.

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

8.10.4.2 clearSinks()

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

Definition at line 51 of file LoggerHandler.cpp.

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

8.10.4.3 logger()

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

Definition at line 43 of file LoggerHandler.cpp.

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

8.10.4.4 setName()

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

Definition at line 14 of file LoggerHandler.cpp.

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

8.10.4.5 setVerbosity()

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

Definition at line 22 of file LoggerHandler.cpp.

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

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

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

8.11 yaodaq::Looper Class Reference

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

Public Member Functions

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

Static Public Member Functions

- static int getInstance ()

8.11.1 Detailed Description

Definition at line 15 of file Looper.hpp.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 Looper()

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

Definition at line 28 of file Looper.cpp.

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

8.11.2.2 ~Looper()

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

Definition at line 52 of file Looper.cpp.

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

8.11.3 Member Function Documentation

8.11.3.1 getInstance()

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

Definition at line 17 of file Looper.cpp.

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

8.11.3.2 getSignal()

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

Definition at line 50 of file Looper.cpp.

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

8.11.3.3 loop()

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

Definition at line 37 of file Looper.cpp.

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

8.11.3.4 supressInstance()

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

Definition at line 19 of file Looper.cpp.

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

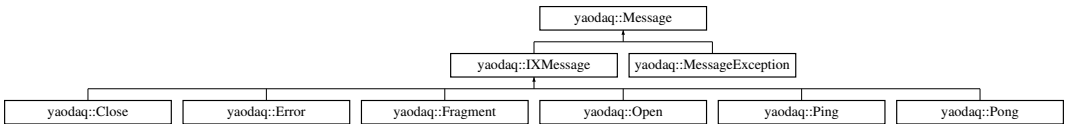
The documentation for this class was generated from the following files:

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

8.12 yaodaq::Message Class Reference

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

Inheritance diagram for yaodaq::Message:



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.12.1 Detailed Description

Definition at line 19 of file Message.hpp.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 Message() [1/5]

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

Definition at line 25 of file Message.cpp.

```
00026 {
00027   m_JSON["from"];
00028   m_JSON["to"];
00029   m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
00030   m_JSON["uuid"] = ix::uuid4();
00031   m_JSON["content"];
00032   m_JSON["timestamp"]                        = fmt::format( "{:%F %T %z}", fmt::gmtime(
      std::chrono::system_clock::to_time_t( std::chrono::system_clock::now() ) ) );
00033   m_JSON["meta"]["compiler"]                 = nlohmann::json::meta()["compiler"];
00034   m_JSON["meta"]["platform"]                 = nlohmann::json::meta()["platform"];
00035   m_JSON["meta"]["versions"]["json"]         = nlohmann::json::meta()["version"]["string"];
00036   m_JSON["meta"]["versions"]["yaodaq"]       = yaodaq_version.to_string();
00037   m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
00038 }
```

8.12.2.2 Message() [2/5]

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

Definition at line 54 of file Message.cpp.

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

8.12.2.3 Message() [3/5]

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

Definition at line 56 of file Message.cpp.

```
00056 : Message( messageType ) { setContent( content ); }
```

8.12.2.4 Message() [4/5]

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

Definition at line 58 of file Message.cpp.

```
00058 : Message( messageType ) { setContent( content ); }
```

8.12.2.5 Message() [5/5]

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

Definition at line 102 of file Message.cpp.

```
00102 : Message() { m_JSON["type"] = magic_enum::enum_name( messageType ); }
```

8.12.3 Member Function Documentation

8.12.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.12.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.12.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.12.3.4 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.12.3.5 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.12.3.6 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.12.3.7 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.12.3.8 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.12.3.9 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.12.3.10 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.12.3.11 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.12.3.12 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.12.4 Field Documentation

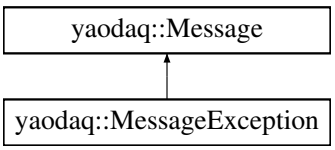
8.12.4.1 m_JSON

`nlohmann::json yaodaq::Message::m_JSON [protected]`
Definition at line 41 of file Message.hpp.
The documentation for this class was generated from the following files:

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

8.13 yaodaq::MessageException Class Reference

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



Public Member Functions

- MessageException (const Exception &content)
- std::int_least32_t getCode ()
- std::string getDescription ()
- std::int_least32_t getLine ()
- std::int_least32_t getColumn ()
- std::string getFileName ()
- std::string getFunctionName ()
- std::string dump (const int &indent=-1, const char &indent_char=' ', const bool &ensure_ascii=false, const nlohmann::detail::error_handler_t &error_handler=nlohmann::detail::error_handler_t::strict) const
- nlohmann::json get () const
- nlohmann::json getContent () const
- std::string getTypeName () const
- MessageType getTypeValue () const
- std::string getTimestamp () const
- std::time_t getTime () const
- Identifier getIdentifier () const
- void setFrom (const Identifier &)

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.13.1 Detailed Description

Definition at line 44 of file Message.hpp.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 MessageException()

```
yaodaq::MessageException::MessageException (
            const Exception & content )  [explicit]
```

Definition at line 105 of file Message.cpp.

```
00105                                                          : Message( MessageType::Exception )
00106 {
00107   nlohmann::json j;
00108   j["code"]          = exception.code();
00109   j["description"]   = exception.description();
00110   j["line"]          = exception.line();
00111   j["column"]        = exception.column();
00112   j["file_name"]     = exception.file_name();
00113   j["function_name"] = exception.function_name();
00114   setContent( j );
00115 }
```

8.13.3 Member Function Documentation

8.13.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.13.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.13.3.3 getCode()

```
std::int_least32_t yaodaq::MessageException::getCode ( )
```

Definition at line 117 of file Message.cpp.

```
00117 { return get()["content"]["code"].get<std::int_least32_t>(); }
```

8.13.3.4 getColumn()

```
std::int_least32_t yaodaq::MessageException::getColumn ( )
```

Definition at line 123 of file Message.cpp.

```
00123 { return get()["content"]["column"].get<std::int_least32_t>(); }
```

8.13.3.5 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.13.3.6 getDescription()

```
std::string yaodaq::MessageException::getDescription ( )
```

Definition at line 119 of file Message.cpp.

```
00119 { return get()["content"]["description"].get<std::string>(); }
```

8.13.3.7 getFileName()

```
std::string yaodaq::MessageException::getFileName ( )
```

Definition at line 125 of file Message.cpp.

```
00125 { return get()["content"]["file_name"].get<std::string>(); }
```

8.13.3.8 getFunctionName()

```
std::string yaodaq::MessageException::getFunctionName ( )
```

Definition at line 127 of file Message.cpp.

```
00127 { return get()["content"]["function_name"].get<std::string>(); }
```

8.13.3.9 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.13.3.10 getLine()

```
std::int_least32_t yaodaq::MessageException::getLine ( )
```

Definition at line 121 of file Message.cpp.

```
00121 { return get()["content"]["line"].get<std::int_least32_t>(); }
```

8.13.3.11 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.13.3.12 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.13.3.13 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.13.3.14 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.13.3.15 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.13.3.16 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.13.3.17 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.13.3.18 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.13.4 Field Documentation

8.13.4.1 m_JSON

```
nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]
```

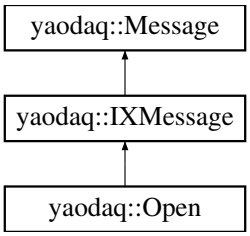
Definition at line 41 of file Message.hpp.

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

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

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.14.1 Detailed Description

Definition at line 34 of file IXWebsocketMessage.hpp.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 Open() [1/2]

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

Definition at line 38 of file IXWebsocketMessage.cpp.

```
00038                                                 : IXMessage( MessageType::Open )
00039 {
00040   nlohmann::json j = getContent();
00041   j["uri"]         = openInfo.uri;
00042   j["headers"]     = openInfo.headers;
00043   j["protocol"]    = openInfo.protocol;
00044   setContent( j );
00045 }
```

8.14.2.2 Open() [2/2]

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

Definition at line 47 of file IXWebsocketMessage.cpp.

```
00047 : Open( openInfo ) { setConnectionStateInfos( connectionState ); }
```

8.14.3 Member Function Documentation

8.14.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.14.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.14.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.14.3.4 getHeaders()

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

Definition at line 51 of file IXWebsocketMessage.cpp.

```
00052 {
00053   std::map<std::string, std::string> ret = get()["content"]["headers"].get<std::map<std::string,
       std::string»();
00054   return ret;
00055 }
```

8.14.3.5 getId()

```
std::string yaodaq::IXMessage::getId ( ) const  [inherited]
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 { return get()["content"]["id"].get<std::string>(); }
```

8.14.3.6 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.14.3.7 getProtocol()

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

Definition at line 57 of file IXWebsocketMessage.cpp.

```
00057 { return get()["content"]["protocol"].get<std::string>(); }
```

8.14.3.8 getRemoteIp()

```
std::string yaodaq::IXMessage::getRemoteIp ( ) const  [inherited]
```

Definition at line 33 of file IXWebsocketMessage.cpp.

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

8.14.3.9 getRemotePort()

```
int yaodaq::IXMessage::getRemotePort ( ) const  [inherited]
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00035 { return get()["content"]["remote_port"].get<int>(); }
```

8.14.3.10 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.14.3.11 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.14.3.12 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.14.3.13 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.14.3.14 getURI()

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

Definition at line 49 of file IXWebsocketMessage.cpp.

```
00049 { return get()["content"]["uri"].get<std::string>(); }
```

8.14.3.15 setConnectionStateInfos()

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

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
```

8.14.3.16 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.14.3.17 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.14.3.18 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.14.3.19 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.14.4 Field Documentation

8.14.4.1 m_JSON

```
nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]
```

Definition at line 41 of file Message.hpp.

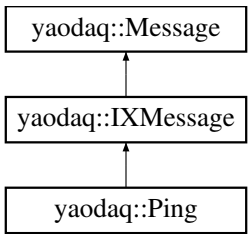
The documentation for this class was generated from the following files:

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

8.15 yaodaq::Ping Class Reference

`#include <yaodaq/IXWebsocketMessage.hpp>`

Inheritance diagram for yaodaq::Ping:



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.15.1 Detailed Description

Definition at line 66 of file IXWebsocketMessage.hpp.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 Ping() [1/2]

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

Definition at line 100 of file IXWebsocketMessage.cpp.

```
00100 : IXMessage( MessageType::Ping ) {}
```

8.15.2.2 Ping() [2/2]

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

Definition at line 102 of file IXWebsocketMessage.cpp.

```
00102 : Ping( ping ) { setConnectionStateInfos( connectionState ); }
```

8.15.3 Member Function Documentation

8.15.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.15.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.15.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.15.3.4 getId()

```
std::string yaodaq::IXMessage::getId ( ) const  [inherited]
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 { return get()["content"]["id"].get<std::string>(); }
```

8.15.3.5 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.15.3.6 getRemoteIp()

```
std::string yaodaq::IXMessage::getRemoteIp ( ) const  [inherited]
```

Definition at line 33 of file IXWebsocketMessage.cpp.

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

8.15.3.7 getRemotePort()

```
int yaodaq::IXMessage::getRemotePort ( ) const  [inherited]
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00035 { return get()["content"]["remote_port"].get<int>(); }
```

8.15.3.8 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.15.3.9 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.15.3.10 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.15.3.11 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.15.3.12 setConnectionStateInfos()

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

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
```

8.15.3.13 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.15.3.14 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.15.3.15 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.15.3.16 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
```

```
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.15.4 Field Documentation

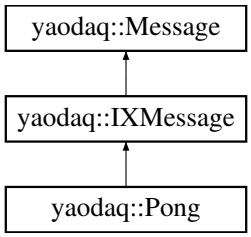
8.15.4.1 m_JSON

`nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]`
Definition at line 41 of file Message.hpp.
The documentation for this class was generated from the following files:

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

8.16 yaodaq::Pong Class Reference

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



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

- nlohmann::json m_JSON

8.16.1 Detailed Description

Definition at line 73 of file IXWebsocketMessage.hpp.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 Pong() [1/2]

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

Definition at line 105 of file IXWebsocketMessage.cpp.

```
00105 : IXMessage( MessageType::Pong ) {}
```

8.16.2.2 Pong() [2/2]

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

Definition at line 107 of file IXWebsocketMessage.cpp.

```
00107 : Pong( pong ) { setConnectionStateInfos( connectionState ); }
```

8.16.3 Member Function Documentation

8.16.3.1 dump()

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

Definition at line 60 of file Message.cpp.

```
00060 { return m_JSON.dump( indent, indent_char, ensure_ascii, error_handler ); }
```

8.16.3.2 get()

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

Definition at line 62 of file Message.cpp.

```
00062 { return m_JSON; }
```

8.16.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
00068 { return m_JSON["content"]; }
```

8.16.3.4 getId()

```
std::string yaodaq::IXMessage::getId ( ) const  [inherited]
```

Definition at line 31 of file IXWebsocketMessage.cpp.

```
00031 { return get()["content"]["id"].get<std::string>(); }
```

8.16.3.5 getIdentifier()

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

Definition at line 90 of file Message.cpp.

```
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
```

8.16.3.6 getRemoteIp()

```
std::string yaodaq::IXMessage::getRemoteIp ( ) const  [inherited]
```

Definition at line 33 of file IXWebsocketMessage.cpp.

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

8.16.3.7 getRemotePort()

```
int yaodaq::IXMessage::getRemotePort ( ) const  [inherited]
```

Definition at line 35 of file IXWebsocketMessage.cpp.

```
00035 { return get()["content"]["remote_port"].get<int>(); }
```

8.16.3.8 getTime()

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

Definition at line 72 of file Message.cpp.

```
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
```

8.16.3.9 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
00070 { return m_JSON["timestamp"].get<std::string>(); }
```

8.16.3.10 getTypeName()

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

Definition at line 64 of file Message.cpp.

```
00064 { return m_JSON["type"].get<std::string>(); }
```

8.16.3.11 getTypeValue()

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

Definition at line 66 of file Message.cpp.

```
00066 { return magic_enum::enum_cast<MessageType>( m_JSON["type"].get<std::string>() ).value(); }
```

8.16.3.12 setConnectionStateInfos()

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

Definition at line 22 of file IXWebsocketMessage.cpp.

```
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
```

8.16.3.13 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

```
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
```

8.16.3.14 setContent() [2/3]

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

Definition at line 40 of file Message.cpp.

```
00040 { m_JSON["content"] = static_cast<nlohmann::json>( content ); }
```

8.16.3.15 setContent() [3/3]

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

Definition at line 42 of file Message.cpp.

```
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
```

8.16.3.16 setFrom()

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

Definition at line 81 of file Message.cpp.

```
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
```

8.16.4 Field Documentation

8.16.4.1 m_JSON

```
nlohmann::json yaodaq::Message::m_JSON  [protected], [inherited]
```

Definition at line 41 of file Message.hpp.

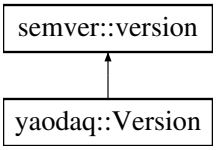
The documentation for this class was generated from the following files:

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

8.17 yaodaq::Version Class Reference

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

Inheritance diagram for yaodaq::Version:



Public Member Functions

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

8.17.1 Detailed Description

Definition at line 15 of file Version.hpp.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 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 ) {}
```

8.17.2.2 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 ) {}
```

8.17.2.3 Version() [3/3]

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

8.17.3 Member Function Documentation

8.17.3.1 getMajor()

```
std::uint8_t yaodaq::Version::getMajor ( )
```
Definition at line 12 of file Version.cpp.
```
00012 { return major; }
```

8.17.3.2 getMinor()

```
std::uint8_t yaodaq::Version::getMinor ( )
```
Definition at line 14 of file Version.cpp.
```
00014 { return minor; }
```

8.17.3.3 getPatch()

```
std::uint8_t yaodaq::Version::getPatch ( )
```
Definition at line 16 of file Version.cpp.
```
00016 { return patch; }
```

8.17.3.4 getPreRelease()

```
std::string yaodaq::Version::getPreRelease ( )
```
Definition at line 18 of file Version.cpp.
```
00018 { return std::string( magic_enum::enum_name( prerelease_type ) ); }
```

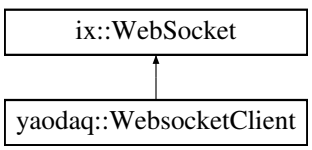
8.17.3.5 getPreReleaseNumber()

```
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

8.18 yaodaq::WebsocketClient Class Reference

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



Public Member Functions

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

Static Public Member Functions

- static void throwGeneralIfSameName (const bool &)

8.18.1 Detailed Description

Definition at line 29 of file WebsocketClient.hpp.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 WebsocketClient()

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

Definition at line 25 of file WebsocketClient.cpp.

```
00025                                                                    : m_Identifier( type,
       name )
00026 {
00027   ix::initNetSystem();
00028
00029   m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
00030   m_Logger.setName( m_Identifier.get() );
00031   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00032
00033   ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
00034   setExtraHeaders( header );
00035
00036   setOnMessageCallback(
00037     [this]( const ix::WebSocketMessagePtr& msg )
00038     {
00039       if( msg->type == ix::WebSocketMessageType::Message )
00040       {
00041         IXMessage ixmessage( msg );
00042         onMessage( ixmessage );
00043       }
00044       else if( msg->type == ix::WebSocketMessageType::Open )
00045       {
00046         Open open( msg->openInfo );
00047         onOpen( open );
00048       }
00049       else if( msg->type == ix::WebSocketMessageType::Close )
00050       {
00051         Close close( msg->closeInfo );
00052         onClose( close );
00053       }
00054       else if( msg->type == ix::WebSocketMessageType::Error )
00055       {
00056         Error error( msg->errorInfo );
00057         onError( error );
00058       }
```

```
00059       else if( msg->type == ix::WebSocketMessageType::Ping )
00060       {
00061         Ping ping( msg );
00062         onPing( ping );
00063       }
00064       else if( msg->type == ix::WebSocketMessageType::Pong )
00065       {
00066         Pong pong( msg );
00067         onPong( pong );
00068       }
00069       else if( msg->type == ix::WebSocketMessageType::Fragment )
00070       {
00071         Fragment fragment( msg );
00072         onFragment( fragment );
00073       }
00074     } );
00075 }
```

8.18.2.2 ~WebsocketClient()

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

Definition at line 123 of file WebsocketClient.cpp.

```
00124 {
00125   stop();
00126   ix::uninitNetSystem();
00127 }
```

8.18.3 Member Function Documentation

8.18.3.1 logger()

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

Definition at line 37 of file WebsocketClient.hpp.

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

8.18.3.2 loop()

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

Definition at line 148 of file WebsocketClient.cpp.

```
00149 {
00150   WebsocketClient::start();
00151   m_Looper.supressInstance();
00152   onRaisingSignal();
00153 }
```

8.18.3.3 onClose()

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

Definition at line 104 of file WebsocketClient.cpp.

```
00104 { logger()->debug( fmt::format( fg( fmt::color::green ), "Connection closed:\nCode: {}\nReason:
      {}\nRemote: {}", close.getCode(), close.getReason(), close.getRemote() ) ); }
```

8.18.3.4 onError()

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

Definition at line 106 of file WebsocketClient.cpp.

```
00107 {
00108   logger()->error( fmt::format( fg( fmt::color::red ), "Error:\nRetries: {}\nWait time: {}\nHTTP
      status: {}\nReason: {}\nCompression error: {}", error.getRetries(), error.getWaitTime(),
      error.getHttpStatus(), error.getDecompressionError() ) );
00109 }
```

8.18.3.5 onException()

```
void yaodaq::WebsocketClient::onException (
            MessageException & message )  [virtual]
```

Definition at line 117 of file WebsocketClient.cpp.

```
00118 {
00119   Exception exception( StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED, message.getDescription()
      );
00120   logger()->critical( "{Exception }", exception.what() );
00121 }
```

8.18.3.6 onFragment()

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

Definition at line 115 of file WebsocketClient.cpp.

```
00115 {}
```

8.18.3.7 onMessage()

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

Definition at line 77 of file WebsocketClient.cpp.

```
00078 {
00079   switch( message.getTypeValue() )
00080   {
00081     case MessageType::Exception:
00082       MessageException& message_exception = reinterpret_cast<MessageException&>( message );
00083       // Special case for connection to the server with the same name as an other client !
00084       if( static_cast<StatusCode>( message_exception.getCode() ) ==
      StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED )
00085       {
00086         disableAutomaticReconnection();
00087         std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
00088         close( message_exception.getCode(), message_exception.getDescription() );
00089         std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
00090         if( m_ThrowGeneralIfSameName ) throw Exception( static_cast<StatusCode>(
      message_exception.getCode() ), message_exception.getDescription() );
00091       }
00092       onException( message_exception );
00093       break;
00094   }
00095 }
```

8.18.3.8 onOpen()

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

Definition at line 97 of file WebsocketClient.cpp.

```
00098 {
00099   std::string headers;
00100   for( auto it: open.getHeaders() ) { headers += fmt::format( "\t{}: {}\n", it.first, it.second ); }
00101   logger()->debug( fmt::format( fg( fmt::color::green ), "Connection opened:\nURI: {}\nProtocol:
      {}\nHeaders:\n{}", open.getURI(), open.getProtocol(), headers ) );
00102 }
```

8.18.3.9 onPing()

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

Definition at line 111 of file WebsocketClient.cpp.

```
00111 { logger()->debug( fmt::format( fg( fmt::color::green ), "Ping:\n{}", ping.getContent().dump( 2 ) ) );
      }
```

8.18.3.10 onPong()

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

Definition at line 113 of file WebsocketClient.cpp.

```
00113 { logger()->debug( fmt::format( fg( fmt::color::green ), "Pong:\n{}", pong.getContent().dump( 2 ) ) );
       }
```

8.18.3.11 start()

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

Definition at line 129 of file WebsocketClient.cpp.

```
00130 {
00131   if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00132   {
00133     logger()->trace( "Client started. Connected to {}", getUrl() );
00134     ix::WebSocket::start();
00135   }
00136 }
```

8.18.3.12 stop()

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

Definition at line 138 of file WebsocketClient.cpp.

```
00139 {
00140   if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00141   {
00142     logger()->trace( "Client stopped" );
00143     ix::WebSocket::stop();
00144     while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
       std::chrono::microseconds( 1 ) ); }
00145   }
00146 }
```

8.18.3.13 throwGeneralIfSameName()

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

Definition at line 23 of file WebsocketClient.cpp.

```
00023 { m_ThrowGeneralIfSameName = activate; }
```

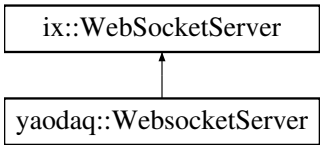
The documentation for this class was generated from the following files:

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

8.19 yaodaq::WebsocketServer Class Reference

`#include <yaodaq/WebsocketServer.hpp>`

Inheritance diagram for yaodaq::WebsocketServer:



Public Member Functions

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

std::size_t &maxConnections=ix::SocketServer::kDefaultMaxConnections, const int &handshakeTimeout↩
Secs=ix::WebSocketServer::kDefaultHandShakeTimeoutSecs, const int &addressFamily=ix::SocketServer↩
::kDefaultAddressFamily, const std::string &type="YAODAQWebsocketServer")

- virtual ~WebsocketServer ()
- void loop ()
- void start ()
- void stop (bool useless=true)
- void listen ()
- virtual void onMessage (Message &message)
- virtual void onOpen (Open &open)
- virtual void onClose (Close &close)
- virtual void onError (Error &error)
- virtual void onPing (Ping &ping)
- virtual void onPong (Pong &pong)
- virtual void onFragment (Fragment &fragment)
- virtual void onException (MessageException &)
- virtual void onUnknown (Message &)
- void setVerbosity (const yaodaq::LoggerHandler::Verbosity &verbosity)
- std::shared_ptr< spdlog::logger > logger ()
- void sendToLoggers (Message &message)
- void sendToLoggers (const Message &message)
- void sendToLoggers (Message &message, ix::WebSocket &webSocket)
- void sendToLoggers (const Message &message, ix::WebSocket &webSocket)

8.19.1 Detailed Description

Definition at line 31 of file WebsocketServer.hpp.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 WebsocketServer()

```
yaodaq::WebsocketServer::WebsocketServer (
            const std::string & name,
            const int & port = ix::SocketServer::kDefaultPort,
            const std::string & host = ix::SocketServer::kDefaultHost,
            const int & backlog = ix::SocketServer::kDefaultTcpBacklog,
            const std::size_t & maxConnections = ix::SocketServer::kDefaultMaxConnections,
            const int & handshakeTimeoutSecs = ix::WebSocketServer::kDefaultHandShakeTimeoutSecs,
            const int & addressFamily = ix::SocketServer::kDefaultAddressFamily,
            const std::string & type = "YAODAQWebsocketServer" )  [explicit]
```

Definition at line 27 of file WebsocketServer.cpp.

```
00027
                                :
00028   ix::WebSocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
       m_Identifier( type, name )
00029 {
00030   ix::initNetSystem();
00031
00032   m_Identifier.generateKey( Domain::Application, Class::Server, Family::WebSocketServer );
00033   m_Logger.setName( m_Identifier.get() );
00034   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00035
00036   setConnectionStateFactory( []() { return std::make_shared<ConnectionState>(); } );
00037
00038   setOnClientMessageCallback(
00039     [this]( std::shared_ptr<ix::ConnectionState> connectionState, ix::WebSocket& webSocket, const
      ix::WebSocketMessagePtr& msg )
00040     {
00041       // The ConnectionState object contains information about the connection
```

```
00042       std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>(
    connectionState );
00043       if( msg->type == ix::WebSocketMessageType::Message )
00044       {
00045         IXMessage ixmessage( msg );
00046         // sendToLoggers(ixmessage,webSocket);
00047         onMessage( ixmessage );
00048       }
00049       else if( msg->type == ix::WebSocketMessageType::Open )
00050       {
00051         // Check if a client with the same name is already connected;
00052         connection->computeId( getHost() + ":" + std::to_string( getPort() ), Identifier::parse(
    msg->openInfo.headers["id"] ) );
00053         if( connection->isTerminated() )
00054         {
00055           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() )
    );
00056           MessageException message_exception(
00057             Exception( StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED, fmt::format( "One client
    with the name \"{}\" is already connected to ws{}://{}:{} !", Identifier::parse(
    msg->openInfo.headers["id"] ).getName(), "", getHost(), getPort() ) ) );
00058
00059           // Send to the client on exception;
00060           webSocket.send( message_exception.dump() );
00061           // Send to loggers except the client as it has been send before;
00062           // sendToLoggers(message_exception,webSocket);
00063
00064           webSocket.stop( magic_enum::enum_integer(
    StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ),
00065                           fmt::format( "One client with the name \"{}\" is already connected to
    ws{}://{}:{} !", Identifier::parse( msg->openInfo.headers["id"] ).getName(), "", getHost(), getPort()
    ) );
00066           std::this_thread::sleep_for( std::chrono::milliseconds( 50 ) );
00067           return;
00068         }
00069         addClient( Identifier::parse( msg->openInfo.headers["id"] ), webSocket );
00070         Open open( msg->openInfo, connection );
00071         // sendToLoggers( open, webSocket );
00072         onOpen( open );
00073       }
00074       else if( msg->type == ix::WebSocketMessageType::Close )
00075       {
00076         Close close( msg->closeInfo, connection );
00077         // sendToLoggers( close, webSocket );
00078         onClose( close );
00079         removeClient( webSocket );
00080       }
00081       else if( msg->type == ix::WebSocketMessageType::Error )
00082       {
00083         Error error( msg->errorInfo, connection );
00084         // sendToLoggers( error, webSocket );
00085         onError( error );
00086       }
00087       else if( msg->type == ix::WebSocketMessageType::Ping )
00088       {
00089         Ping ping( msg, connection );
00090         // sendToLoggers( ping, webSocket );
00091         onPing( ping );
00092       }
00093       else if( msg->type == ix::WebSocketMessageType::Pong )
00094       {
00095         Pong pong( msg, connection );
00096         // sendToLoggers( pong, webSocket );
00097         onPong( pong );
00098       }
00099       else if( msg->type == ix::WebSocketMessageType::Fragment )
00100       {
00101         Fragment fragment( msg, connection );
00102         // sendToLoggers( fragment, webSocket );
00103         onFragment( fragment );
00104       }
00105     } );
00106 }
```

8.19.2.2 ~WebsocketServer()

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

Definition at line 245 of file WebsocketServer.cpp.

```
00246 {
00247   stop();
00248   ix::uninitNetSystem();
00249 }
```

8.19.3 Member Function Documentation

8.19.3.1 listen()

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

Definition at line 207 of file WebsocketServer.cpp.

```
00208 {
00209   if( !m_isListening )
00210   {
00211     std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
00212     if( ret.first )
00213     {
00214       m_isListening = ret.first;
00215       logger()->info( "Server listening on {0}:{1}", getHost(), getPort() );
00216     }
00217     else
00218       throw Exception( StatusCode::LISTEN_ERROR, ret.second );
00219   }
00220 }
```

8.19.3.2 logger()

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

Definition at line 57 of file WebsocketServer.hpp.

```
00057 { return m_Logger.logger(); }
```

8.19.3.3 loop()

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

Definition at line 251 of file WebsocketServer.cpp.

```
00252 {
00253   listen();
00254   start();
00255   m_Looper.supressInstance();
00256   onRaisingSignal();
00257 }
```

8.19.3.4 onClose()

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

Definition at line 182 of file WebsocketServer.cpp.

```
00183 {
00184   logger()->debug(
00185     fmt::format( fg( fmt::color::green ), "Connection closed:\nId: {}\nRemote IP: {}\nRemote port:
      {}\nCode: {}\nReason: {}\nRemote: {}", close.getId(), close.getRemoteIp(), close.getRemotePort(),
      close.getCode(), close.getReason(), close.getRemote() ) );
00186 }
```

8.19.3.5 onError()

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

Definition at line 188 of file WebsocketServer.cpp.

```
00189 {
00190   logger()->error( fmt::format( fg( fmt::color::red ), "Error:\nId: {}\nRemote IP: {}\nRemote port:
      {}\nRetries: {}\nWait time: {}\nHTTP status: {}\nReason: {}\nCompression error: {}", error.getId(),
      error.getRemoteIp(), error.getRemotePort(),
00191                             error.getRetries(), error.getWaitTime(), error.getHttpStatus(),
      error.getDecompressionError() ) );
00192 }
```

8.19.3.6 onException()

```
void yaodaq::WebsocketServer::onException (
            MessageException & message )  [virtual]
```

Definition at line 200 of file WebsocketServer.cpp.

```
00200 {}
```

8.19.3.7 onFragment()

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

Definition at line 198 of file WebsocketServer.cpp.

```
00198 {}
```

8.19.3.8 onMessage()

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

Definition at line 159 of file WebsocketServer.cpp.

```
00160 {
00161   switch( message.getTypeValue() )
00162   {
00163     case MessageType::Exception:
00164     {
00165       MessageException& message_exception = reinterpret_cast<MessageException&>( message );
00166       onException( message_exception );
00167     }
00168     default:
00169     {
00170       onUnknown( message );
00171     }
00172   }
00173 }
```

8.19.3.9 onOpen()

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

Definition at line 175 of file WebsocketServer.cpp.

```
00176 {
00177   std::string headers;
00178   for( auto it: open.getHeaders() ) { headers += fmt::format( "\t{}: {}\n", it.first, it.second ); }
00179   logger()->debug( fmt::format( fg( fmt::color::green ), "Connection opened:\nId: {}\nRemote IP:
      {}\nRemote port: {}\nURI: {}\nProtocol: {}\nHeaders:\n{}", open.getId(), open.getRemoteIp(),
      open.getRemotePort(), open.getURI(), open.getProtocol(), headers ) );
00180 }
```

8.19.3.10 onPing()

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

Definition at line 194 of file WebsocketServer.cpp.

```
00194 { logger()->debug( fmt::format( fg( fmt::color::green ), "Ping:\n{}", ping.getContent().dump( 2 ) ) );
      }
```

8.19.3.11 onPong()

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

Definition at line 196 of file WebsocketServer.cpp.

```
00196 { logger()->debug( fmt::format( fg( fmt::color::green ), "Pong:\n{}", pong.getContent().dump( 2 ) ) );
      }
```

8.19.3.12 onUnknown()

```
void yaodaq::WebsocketServer::onUnknown (
            Message & unknown )  [virtual]
```

Definition at line 202 of file WebsocketServer.cpp.

```
00203 {
00204   logger()->error( fmt::format( fg( fmt::color::red ), "Unknwown:\n{}", unknown.dump( 2 ) ) );
00205 }
```

8.19.3.13 sendToLoggers() [1/4]

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

Definition at line 151 of file WebsocketServer.cpp.

```
00152 {
00153   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00154   {
00155     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
      it->second.send( message.dump() );
00156   }
00157 }
```

8.19.3.14 sendToLoggers() [2/4]

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

Definition at line 143 of file WebsocketServer.cpp.

```
00144 {
00145   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00146   {
00147     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
      != &it->second ) it->second.send( message.dump() );
00148   }
00149 }
```

8.19.3.15 sendToLoggers() [3/4]

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

Definition at line 135 of file WebsocketServer.cpp.

```
00136 {
00137   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00138   {
00139     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
      it->second.send( message.dump() );
00140   }
00141 }
```

8.19.3.16 sendToLoggers() [4/4]

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

Definition at line 127 of file WebsocketServer.cpp.

```
00128 {
00129   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00130   {
00131     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
      != &it->second ) it->second.send( message.dump() );
00132   }
00133 }
```

8.19.3.17 setVerbosity()

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

Definition at line 243 of file WebsocketServer.cpp.

```
00243 { m_Logger.setVerbosity( verbosity ); }
```

8.19.3.18 start()

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

Definition at line 222 of file WebsocketServer.cpp.

```
00223 {
00224   if( !m_isStarted )
00225   {
00226     m_isStarted = true;
00227     logger()->trace( "Server started" );
00228     ix::WebSocketServer::start();
00229   }
00230 }
```

8.19.3.19 stop()

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

Definition at line 232 of file WebsocketServer.cpp.

```
00233 {
00234   if( !m_isStopped )
00235   {
00236     m_isStopped = true;
00237     useless     = !useless;
00238     logger()->trace( "Server stopped" );
00239     ix::WebSocketServer::stop();
00240   }
00241 }
```

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

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

Chapter 9

File Documentation

9.1 docs/License.md File Reference

9.2 docs/Third-party licenses.md File Reference

9.3 yaodaq/Classification.hpp File Reference

```
#include <cstdint>
```

Namespaces

- namespace yaodaq

Enumerations

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

9.4 Classification.hpp

Go to the documentation of this file.

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

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

9.5 yaodaq/ConnectionState.hpp File Reference

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

Data Structures

- class yaodaq::ConnectionState

Namespaces

- namespace yaodaq

9.6 ConnectionState.hpp

Go to the documentation of this file.

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

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

9.7 yaodaq/Exception.hpp File Reference

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

Data Structures

- class yaodaq::Exception

Namespaces

- namespace yaodaq

9.8 Exception.hpp

Go to the documentation of this file.

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

9.9 yaodaq/Identifier.hpp File Reference

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

Data Structures

- class yaodaq::Identifier

Namespaces

- namespace yaodaq

9.10 Identifier.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_IDENTIFIER
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_lass =
       Class::Unknown, const Family& family = Family::Unknown );
00022   [[nodiscard]] std::string getDomain() const;
00023   [[nodiscard]] std::string getClass() const;
00024   [[nodiscard]] std::string getFamily() const;
00025   [[nodiscard]] std::string getType() const;
00026   [[nodiscard]] std::string getName() const;
00027   [[nodiscard]] Key         getKey() const;
00028   [[nodiscard]] std::string get() const;
00029   bool                    empty() const;
00030   static Identifier       parse( const std::string& );
00031   bool                    operator<( const Identifier& ) const;
00032
00033 private:
00034   std::string m_Type{ "Unknown" };
00035   std::string m_Name{ "Unknown" };
00036   Key         m_Key;
00037 };
00038
00039 }  // namespace yaodaq
00040
00041 #endif  // YAODAQ_IDENTIFIER
```

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/IXWebSocketCloseInfo.h>
#include <ixwebsocket/IXWebSocketErrorInfo.h>
#include <ixwebsocket/IXWebSocketMessage.h>
#include <ixwebsocket/IXWebSocketOpenInfo.h>
#include <map>
#include <memory>
#include <string>
```

Data Structures

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

Namespaces

- namespace yaodaq

9.14 IXWebsocketMessage.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_IXWEBSOCKETMESSAGE
00002 #define YAODAQ_IXWEBSOCKETMESSAGE
00003
00008 #include "yaodaq/ConnectionState.hpp"
00009 #include "yaodaq/Message.hpp"
00010
00011 #include <ixwebsocket/IXWebSocketCloseInfo.h>
00012 #include <ixwebsocket/IXWebSocketErrorInfo.h>
00013 #include <ixwebsocket/IXWebSocketMessage.h>
00014 #include <ixwebsocket/IXWebSocketOpenInfo.h>
00015 #include <map>
00016 #include <memory>
00017 #include <string>
00018
00019 namespace yaodaq
00020 {
00021
00022 class IXMessage : public Message
00023 {
00024 public:
00025   explicit IXMessage( const MessageType& messageType );
00026   explicit IXMessage( const ix::WebSocketMessagePtr& msg );
00027   std::string getId() const;
00028   std::string getRemoteIp() const;
00029   int getRemotePort() const;
00030 protected:
00031   void setConnectionStateInfos( std::shared_ptr<ConnectionState>& connectionState );
00032 };
00033
00034 class Open : public IXMessage
00035 {
00036 public:
00037   explicit Open( const ix::WebSocketOpenInfo& openInfo );
00038   Open( const ix::WebSocketOpenInfo& openInfo, std::shared_ptr<ConnectionState>& connectionState );
00039   std::string                        getURI() const;
00040   std::map<std::string, std::string> getHeaders() const;
00041   std::string                        getProtocol() const;
00042 };
00043
00044 class Close : public IXMessage
00045 {
00046 public:
00047   explicit Close( const ix::WebSocketCloseInfo& closeInfo );
00048   Close( const ix::WebSocketCloseInfo& closeInfo, std::shared_ptr<ConnectionState>& connectionState );
00049   std::uint16_t getCode() const;
00050   std::string   getReason() const;
00051   bool          getRemote() const;
00052 };
00053
00054 class Error : public IXMessage
00055 {
00056 public:
00057   explicit Error( const ix::WebSocketErrorInfo& errorInfo );
00058   Error( const ix::WebSocketErrorInfo& errorInfo, std::shared_ptr<ConnectionState>& connectionState );
00059   std::uint16_t getRetries() const;
00060   double        getWaitTime() const;
00061   int           getHttpStatus() const;
00062   std::string   getReason() const;
00063   bool          getDecompressionError() const;
00064 };
00065
00066 class Ping : public IXMessage
00067 {
00068 public:
00069   explicit Ping( const ix::WebSocketMessagePtr& ping );
00070   Ping( const ix::WebSocketMessagePtr& ping, std::shared_ptr<ConnectionState>& connectionState );
00071 };
00072
00073 class Pong : public IXMessage
00074 {
00075 public:
00076   explicit Pong( const ix::WebSocketMessagePtr& pong );
00077   Pong( const ix::WebSocketMessagePtr& pong, std::shared_ptr<ConnectionState>& connectionState );
00078 };
00079
00080 class Fragment : public IXMessage
00081 {
00082 public:
00083   explicit Fragment( const ix::WebSocketMessagePtr& fragment );
00084   Fragment( const ix::WebSocketMessagePtr& fragment, std::shared_ptr<ConnectionState>& connectionState
      );
00085 };
00086
```

```
00087 }  // namespace yaodaq
00088 #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_lass, const Family& family );
00023   [[nodiscard]] std::int_least8_t  getDomain() const;
00024   [[nodiscard]] std::int_least8_t  getClass() const;
00025   [[nodiscard]] std::int_least16_t getFamily() const;
00026   [[nodiscard]] std::int_least32_t getKey() const;
00027 };
00028
00029 }  // namespace yaodaq
00030
00031 #endif  // YAODAQ_KEY
```

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
- class yaodaq::MessageException

Namespaces

- namespace yaodaq

9.22 Message.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_MESSAGE
00002 #define YAODAQ_MESSAGE
00003
00008 #include "nlohmann/json.hpp"
00009 #include "yaodaq/MessageType.hpp"
00010
00011 #include <string>
00012
00013 namespace yaodaq
00014 {
00015
00016 class Identifier;
00017 class Exception;
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   nlohmann::json 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   nlohmann::json m_JSON;
00042 };
00043
00044 class MessageException : public Message
00045 {
00046 public:
00047   explicit MessageException( const Exception& content );
00048   std::int_least32_t getCode();
00049   std::string        getDescription();
00050   std::int_least32_t getLine();
00051   std::int_least32_t getColumn();
00052   std::string        getFileName();
00053   std::string        getFunctionName();
00054 };
00055
00056 }  // namespace yaodaq
00057
00058 #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 = -6 , yaodaq::Close , yaodaq::Error , yaodaq::Ping ,
  yaodaq::Pong , yaodaq::Fragment , yaodaq::Unknown = 0 , yaodaq::Exception }

Functions

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

9.24 MessageType.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_MESSAGETYPE
00002 #define YAODAQ_MESSAGETYPE
00003
00007 #include "yaodaq/Interrupt.hpp"
00008
00009 #include <cstdint>
00010 #include <iosfwd>
00011
00012 namespace yaodaq
00013 {
00014
00015 enum class MessageType : std::int_least16_t
00016 {
00017   // IXWebSocket MessageType (Message is not set here)
00018   Open = -6,
```

```
00019   Close,
00020   Error,
00021   Ping,
00022   Pong,
00023   Fragment,
00024   // Unknown should not be used !
00025   Unknown = 0,
00026   Exception,
00027 };
00028
00029 inline std::ostream& operator«( std::ostream& os, const MessageType& messageTypes ) { return os «
       static_cast<std::int_least8_t>( messageTypes ); }
00030
00031 }  // namespace yaodaq
00032
00033 #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 Message;
00021 class Open;
00022 class Close;
00023 class Error;
00024 class Ping;
00025 class Pong;
00026 class Fragment;
00027 class MessageException;
00028
00029 class WebsocketClient : public ix::WebSocket
00030 {
00031 public:
00032   explicit WebsocketClient( const std::string& name, const std::string& type = "YAODAQWebsocketClient"
      );
00033   virtual ~WebsocketClient();
00034   void                          start();
00035   void                          stop();
00036   void                          loop();
00037   std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
00038   static void throwGeneralIfSameName( const bool& );
00039
00040   // IXWebsocket
00041   virtual void onMessage( Message& message );
00042   virtual void onOpen( Open& open );
00043   virtual void onClose( Close& close );
00044   virtual void onError( Error& error );
00045   virtual void onPing( Ping& ping );
00046   virtual void onPong( Pong& pong );
00047   virtual void onFragment( Fragment& fragment );
00048
00049   virtual void onException(MessageException& );
00050
00051 private:
00052   void          onRaisingSignal();
00053   Identifier    m_Identifier;
00054   LoggerHandler m_Logger;
00055   Looper        m_Looper;
00056   static bool m_ThrowGeneralIfSameName;
00057 };
00058
00059 }  // namespace yaodaq
00060
00061 #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 <map>
```

```
#include <memory>
#include <mutex>
#include <spdlog/spdlog.h>
#include <string>
```

Data Structures

- class yaodaq::WebsocketServer

Namespaces

- namespace yaodaq

9.36 WebsocketServer.hpp

Go to the documentation of this file.

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

```
00061   void sendToLoggers( Message& message, ix::WebSocket& webSocket );
00062   void sendToLoggers( const Message& message, ix::WebSocket& webSocket );
00063
00064 private:
00065   void                                addClient( const Identifier&, ix::WebSocket& );
00066   void                                removeClient( ix::WebSocket& );
00067   void                                onRaisingSignal();
00068   bool                                m_isListening{ false };
00069   Identifier                          m_Identifier;
00070   LoggerHandler                       m_Logger;
00071   Looper                              m_Looper;
00072   bool                                m_isStopped{ false };
00073   bool                                m_isStarted{ false };
00074   std::map<Identifier, ix::WebSocket&> m_Clients;
00075   std::mutex                          m_Mutex;
00076 };
00077
00078 }  // namespace yaodaq
00079
00080 #endif  // YAODAQ_WEBSOCKETSERVER
```

9.37 yaodaq/ConnectionState.cpp File Reference

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

Namespaces

- namespace yaodaq

9.38 ConnectionState.cpp

Go to the documentation of this file.

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

9.39 yaodaq/Exception.cpp File Reference

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

Namespaces

- namespace yaodaq

9.40 Exception.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Exception.hpp"
00006
00007 namespace yaodaq
00008 {
00009
00010 std::string Exception::m_Format{ "\n\t[Code] : {Code}\n\t[Description] : {Description}\n\t[File] :
       {File}\n\t[Function] : {Function}\n\t[Line] : {Line}\n\t[Column] : {Column}\n" };
00011
00012 fmt::text_style Exception::m_Style = { fg( fmt::color::crimson ) | fmt::emphasis::bold };
00013
00014 Exception::Exception( const StatusCode& statusCode, const std::string& description, const
       source_location& location ) : source_location( location ), m_Code( static_cast<std::int_least32_t>(
       statusCode ) ), m_Description( description ) { constructMessage(); }
00015
00016 const char* Exception::what() const noexcept { return m_Message.c_str(); }
00017
00018 const char* Exception::description() const noexcept { return m_Description.c_str(); }
00019
00020 std::int_least32_t Exception::code() const noexcept { return m_Code; }
00021
00022 void Exception::constructMessage()
00023 {
00024   m_Message = fmt::format( m_Style, m_Format, fmt::arg( "Code", m_Code ), fmt::arg( "Description",
       m_Description ), fmt::arg( "File", file_name() ), fmt::arg( "Function", function_name() ), fmt::arg(
       "Column", column() ), fmt::arg( "Line", line() ) );
00025 }
00026
00027 }  // namespace yaodaq
```

9.41 yaodaq/Identifier.cpp File Reference

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

Namespaces

- namespace yaodaq

9.42 Identifier.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Identifier.hpp"
00006
00007 #include "yaodaq/Exception.hpp"
00008 #include "yaodaq/Key.hpp"
00009 #include "yaodaq/StatusCode.hpp"
00010
```

```
00011 #include <fmt/color.h>
00012 #include <magic_enum.hpp>
00013 #include <string>
00014 #include <vector>
00015
00016 namespace yaodaq
00017 {
00018
00019 bool Identifier::empty() const
00020 {
00021   if( get() == Identifier().get() ) return true;
00022   else
00023     return false;
00024 }
00025
00026 Identifier::Identifier( const std::string& type, const std::string& name ) : m_Type( type ), m_Name(
       name ) {}
00027
00028 void Identifier::generateKey( const Domain& domain, const Class& c_lass, const Family& family ) {
       m_Key = Key( domain, c_lass, family ); }
00029
00030 std::string Identifier::getDomain() const { return static_cast<std::string>( magic_enum::enum_name(
       magic_enum::enum_cast<Domain>( m_Key.getDomain() ).value() ) ); }
00031
00032 std::string Identifier::getClass() const { return static_cast<std::string>( magic_enum::enum_name(
       magic_enum::enum_cast<Class>( m_Key.getClass() ).value() ) ); }
00033
00034 std::string Identifier::getFamily() const { return static_cast<std::string>( magic_enum::enum_name(
       magic_enum::enum_cast<Family>( m_Key.getFamily() ).value() ) ); }
00035
00036 std::string Identifier::getType() const { return m_Type; }
00037
00038 std::string Identifier::getName() const { return m_Name; }
00039
00040 Key Identifier::getKey() const { return m_Key; }
00041
00042 std::string Identifier::get() const { return fmt::format( "{0}/{1}/{2}/{3}/{4}", getDomain(),
       getClass(), getFamily(), getType(), getName() ); }
00043
00044 Identifier Identifier::parse( const std::string& id )
00045 {
00046   std::vector<std::string> result;
00047   std::string              tmp        = id;
00048   std::string              separator  = "/";
00049   std::size_t              second_pos = tmp.find( separator );
00050   while( second_pos != std::string::npos )
00051   {
00052     if( 0 != second_pos )
00053     {
00054       std::string word = tmp.substr( 0, second_pos - 0 );
00055       result.push_back( word );
00056     }
00057     else
00058       result.push_back( "" );
00059     tmp        = tmp.substr( second_pos + separator.length() );
00060     second_pos = tmp.find( separator );
00061     if( second_pos == std::string::npos ) result.push_back( tmp );
00062   }
00063   if( result.size() == 5 )
00064   {
00065     Identifier identifier( result[3], result[4] );
00066     identifier.generateKey( magic_enum::enum_cast<Domain>( result[0] ).value(),
       magic_enum::enum_cast<Class>( result[1] ).value(), magic_enum::enum_cast<Family>( result[2] ).value()
       );
00067     return identifier;
00068   }
00069   else
00070   {
00071     throw Exception( StatusCode::WRONG_NUMBER_PARAMETERS, "Number of parameters in key should be 5
       (Domain/Class/Family/Type/Name) !" );
00072   }
00073 }
00074
00075 bool Identifier::operator<( const Identifier& identifier ) const { return this->get() <
       identifier.get(); }
00076
00077 }  // namespace yaodaq
```

9.43 yaodaq/Interrupt.cpp File Reference

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

```
#include <csignal>
#include <mutex>
#include <thread>
```

Namespaces

- namespace yaodaq

9.44 Interrupt.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Interrupt.hpp"
00006
00007 #include "yaodaq/Signal.hpp"
00008
00009 #include <atomic>
00010 #include <csignal>
00011 #include <mutex>
00012 #include <thread>
00013
00014 namespace yaodaq
00015 {
00016
00017 volatile std::atomic<Signal> Interrupt::m_Signal = Signal::NO;
00018
00019 Interrupt::Interrupt() { init(); }
00020
00021 void Interrupt::restore()
00022 {
00023   std::signal( SIGTERM, SIG_DFL );
00024   std::signal( SIGSEGV, SIG_DFL );
00025   std::signal( SIGINT, SIG_DFL );
00026   std::signal( SIGILL, SIG_DFL );
00027   std::signal( SIGABRT, SIG_DFL );
00028   std::signal( SIGFPE, SIG_DFL );
00029 }
00030
00031 void Interrupt::init()
00032 {
00033   setSignal( Signal::TERM );
00034   setSignal( Signal::TERM );
00035   setSignal( Signal::SEGV );
00036   setSignal( Signal::INT );
00037   setSignal( Signal::ILL );
00038   setSignal( Signal::ABRT );
00039   setSignal( Signal::FPE );
00040 }
00041
00042 Interrupt::~Interrupt() { restore(); }
00043
00044 Signal Interrupt::getSignal()
00045 {
00046   if( m_Signal.load() != Signal::NO )
00047   {
00048     std::lock_guard<std::mutex> guard( m_mutex );
00049     init();
00050   }
00051   return m_Signal.load();
00052 }
00053
00054 void Interrupt::setSignal( const Signal& signal )
00055 {
00056   switch( signal )
00057   {
00058     case Signal::ABRT: std::signal( SIGABRT, []( int ) -> void { m_Signal.store( Signal::ABRT ); } );
      break;
00059     case Signal::FPE: std::signal( SIGFPE, []( int ) -> void { m_Signal.store( Signal::FPE ); } );
      break;
00060     case Signal::ILL: std::signal( SIGILL, []( int ) -> void { m_Signal.store( Signal::ILL ); } );
      break;
00061     case Signal::SEGV: std::signal( SIGSEGV, []( int ) -> void { m_Signal.store( Signal::SEGV ); } );
      break;
00062     case Signal::INT: std::signal( SIGINT, []( int ) -> void { m_Signal.store( Signal::INT ); } );
      break;
00063     case Signal::TERM: std::signal( SIGTERM, []( int ) -> void { m_Signal.store( Signal::TERM ); } );
      break;
00064     default: break;
00065   }
00066 }
```

```
00067
00068 }  // namespace yaodaq
```

9.45 yaodaq/IXWebsocketMessage.cpp File Reference

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

Namespaces

- namespace yaodaq

9.46 IXWebsocketMessage.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/IXWebsocketMessage.hpp"
00006
00007 namespace yaodaq
00008 {
00009
00010 IXMessage::IXMessage( const MessageType& messageType ) : Message( messageType ) {}
00011
00012 IXMessage::IXMessage( const ix::WebSocketMessagePtr& msg ) : Message()
00013 {
00014   // FIXME
00015   nlohmann::json json = nlohmann::json::parse( msg->str, nullptr, false );
00016   if( json.is_discarded() ) { m_JSON["content"] = static_cast<std::string>( msg->str ); }
00017   else
00018     m_JSON = json;
00019   std::cout « m_JSON.dump() « std::endl;
00020 }
00021
00022 void IXMessage::setConnectionStateInfos( std::shared_ptr<ConnectionState>& connectionState )
00023 {
00024   nlohmann::json j = getContent();
00025   j["id"]          = connectionState->getId();
00026   j["remote_ip"]   = connectionState->getRemoteIp();
00027   j["remote_port"] = connectionState->getRemotePort();
00028   setContent( j );
00029 }
00030
00031 std::string IXMessage::getId() const { return get()["content"]["id"].get<std::string>(); }
00032
00033 std::string IXMessage::getRemoteIp() const { return get()["content"]["remote_ip"].get<std::string>();
       }
00034
00035 int IXMessage::getRemotePort() const { return get()["content"]["remote_port"].get<int>(); }
00036
00037 // Open
00038 Open::Open( const ix::WebSocketOpenInfo& openInfo ) : IXMessage( MessageType::Open )
00039 {
00040   nlohmann::json j = getContent();
00041   j["uri"]         = openInfo.uri;
00042   j["headers"]     = openInfo.headers;
00043   j["protocol"]    = openInfo.protocol;
00044   setContent( j );
00045 }
00046
00047 Open::Open( const ix::WebSocketOpenInfo& openInfo, std::shared_ptr<ConnectionState>& connectionState )
       : Open( openInfo ) { setConnectionStateInfos( connectionState ); }
00048
00049 std::string Open::getURI() const { return get()["content"]["uri"].get<std::string>(); }
00050
00051 std::map<std::string, std::string> Open::getHeaders() const
00052 {
00053   std::map<std::string, std::string> ret = get()["content"]["headers"].get<std::map<std::string,
      std::string»();
00054   return ret;
00055 }
00056
00057 std::string Open::getProtocol() const { return get()["content"]["protocol"].get<std::string>(); }
00058
00059 // Close
00060 Close::Close( const ix::WebSocketCloseInfo& closeInfo ) : IXMessage( MessageType::Close )
00061 {
00062   nlohmann::json j;
00063   j["code"]   = closeInfo.code;
```

```
00064   j["reason"] = closeInfo.reason;
00065   j["remote"] = closeInfo.remote;
00066   setContent( j );
00067 }
00068
00069 Close::Close( const ix::WebSocketCloseInfo& closeInfo, std::shared_ptr<ConnectionState>&
       connectionState ) : Close( closeInfo ) { setConnectionStateInfos( connectionState ); }
00070
00071 std::uint16_t Close::getCode() const { return get()["content"]["code"].get<std::uint16_t>(); }
00072 std::string   Close::getReason() const { return get()["content"]["reason"].get<std::string>(); }
00073 bool          Close::getRemote() const { return get()["content"]["remote"].get<bool>(); }
00074
00075 // Error
00076 Error::Error( const ix::WebSocketErrorInfo& errorInfo ) : IXMessage( MessageType::Error )
00077 {
00078   nlohmann::json j;
00079   j["retries"]             = errorInfo.retries;
00080   j["wait_time"]           = errorInfo.wait_time;
00081   j["http_status"]         = errorInfo.http_status;
00082   j["reason"]              = errorInfo.reason;
00083   j["decompression_error"] = errorInfo.decompressionError;
00084   setContent( j );
00085 }
00086
00087 Error::Error( const ix::WebSocketErrorInfo& errorInfo, std::shared_ptr<ConnectionState>&
       connectionState ) : Error( errorInfo ) { setConnectionStateInfos( connectionState ); }
00088
00089 std::uint16_t Error::getRetries() const { return get()["content"]["retries"].get<std::uint16_t>(); }
00090
00091 double Error::getWaitTime() const { return get()["content"]["wait_time"].get<double>(); }
00092
00093 int Error::getHttpStatus() const { return get()["content"]["http_status"].get<int>(); }
00094
00095 std::string Error::getReason() const { return get()["content"]["reason"].get<std::string>(); }
00096
00097 bool Error::getDecompressionError() const { return
       get()["content"]["decompression_error"].get<bool>(); }
00098
00099 // Ping
00100 Ping::Ping( const ix::WebSocketMessagePtr& ping ) : IXMessage( MessageType::Ping ) {}
00101
00102 Ping::Ping( const ix::WebSocketMessagePtr& ping, std::shared_ptr<ConnectionState>& connectionState ) :
       Ping( ping ) { setConnectionStateInfos( connectionState ); }
00103
00104 // Pong
00105 Pong::Pong( const ix::WebSocketMessagePtr& pong ) : IXMessage( MessageType::Pong ) {}
00106
00107 Pong::Pong( const ix::WebSocketMessagePtr& pong, std::shared_ptr<ConnectionState>& connectionState ) :
       Pong( pong ) { setConnectionStateInfos( connectionState ); }
00108
00109 // Fragment
00110 Fragment::Fragment( const ix::WebSocketMessagePtr& fragment ) : IXMessage( MessageType::Fragment ) {}
00111
00112 Fragment::Fragment( const ix::WebSocketMessagePtr& fragment, std::shared_ptr<ConnectionState>&
       connectionState ) : Fragment( fragment ) { setConnectionStateInfos( connectionState ); }
00113
00114 }  // namespace yaodaq
```

9.47 yaodaq/Key.cpp File Reference

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

Namespaces

- namespace yaodaq

9.48 Key.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Key.hpp"
00006
00007 #include <cstdint>
00008
00009 namespace yaodaq
00010 {
```

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

9.49 yaodaq/LoggerHandler.cpp File Reference

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

Namespaces

- namespace yaodaq

9.50 LoggerHandler.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/LoggerHandler.hpp"
00006
00007 #include "spdlog/spdlog.h"
00008
00009 namespace yaodaq
00010 {
00011
00012 LoggerHandler::LoggerHandler() { init(); }
00013
00014 void LoggerHandler::setName( const std::string& name )
00015 {
00016   m_Name = name;
00017   init();
00018 }
00019
00020 LoggerHandler::~LoggerHandler() {}
00021
00022 void LoggerHandler::setVerbosity( const Verbosity& verbosity )
00023 {
00024   m_Verbosity = verbosity;
00025   init();
00026 }
00027
00028 void LoggerHandler::init()
00029 {
00030   m_Logger = std::make_shared<spdlog::logger>( m_Name, std::begin( m_Sinks ), std::end( m_Sinks ) );
00031   switch( m_Verbosity )
00032   {
00033     case Verbosity::Off: m_Logger->set_level( spdlog::level::off ); break;
00034     case Verbosity::Trace: m_Logger->set_level( spdlog::level::trace ); break;
00035     case Verbosity::Debug: m_Logger->set_level( spdlog::level::debug ); break;
00036     case Verbosity::Info: m_Logger->set_level( spdlog::level::info ); break;
00037     case Verbosity::Warn: m_Logger->set_level( spdlog::level::warn ); break;
00038     case Verbosity::Error: m_Logger->set_level( spdlog::level::err ); break;
00039     case Verbosity::Critical: m_Logger->set_level( spdlog::level::critical ); break;
00040   }
00041 }
00042
00043 std::shared_ptr<spdlog::logger> LoggerHandler::logger() { return std::shared_ptr<spdlog::logger>(
       m_Logger ); }
00044
00045 void LoggerHandler::addSink( const spdlog::sink_ptr& sink )
00046 {
00047   m_Sinks.push_back( sink );
00048   init();
00049 }
00050
00051 void LoggerHandler::clearSinks()
00052 {
00053   m_Sinks.clear();
```

```
00054   init();
00055 }
00056
00057 }  // namespace yaodaq
```

9.51 yaodaq/Looper.cpp File Reference

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

Namespaces

- namespace yaodaq

9.52 Looper.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Looper.hpp"
00006
00007 #include <chrono>
00008 #include <thread>
00009
00010 namespace yaodaq
00011 {
00012
00013 int Looper::m_instance{ 0 };
00014
00015 Interrupt Looper::m_Interrupt{ Interrupt{} };
00016
00017 int Looper::getInstance() { return m_instance; }
00018
00019 void Looper::supressInstance()
00020 {
00021   if( m_hasBeenSupressed == false )
00022   {
00023     m_hasBeenSupressed = true;
00024     m_instance--;
00025   }
00026 }
00027
00028 Looper::Looper()
00029 {
00030   if( m_hasBeenAdded == false )
00031   {
00032     m_hasBeenAdded = true;
00033     ++m_instance;
00034   }
00035 }
00036
00037 Signal Looper::loop()
00038 {
00039   static Signal signal{ yaodaq::Signal::NO };
00040   if( m_instance == 0 )
00041   {
00042     do {
00043       signal = m_Interrupt.getSignal();
00044       std::this_thread::sleep_for( std::chrono::microseconds( 1 ) );
00045     } while( signal == yaodaq::Signal::NO );
00046   }
00047   return signal;
00048 }
00049
00050 Signal Looper::getSignal() { return m_Interrupt.getSignal(); }
00051
00052 Looper::~Looper()
00053 {
00054   if( m_hasBeenAdded == true && m_hasBeenSupressed == false )
00055   {
00056     m_hasBeenSupressed = true;
00057     --m_instance;
00058   }
00059 }
00060
00061 }  // namespace yaodaq
```

9.53 yaodaq/Message.cpp File Reference

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

Namespaces

- namespace yaodaq

9.54 Message.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Message.hpp"
00006
00007 #include "fmt/chrono.h"
00008 #include "magic_enum.hpp"
00009 #include "yaodaq/Classification.hpp"
00010 #include "yaodaq/Exception.hpp"
00011 #include "yaodaq/Identifier.hpp"
00012
00013 #include <chrono>
00014 #include <ctime>
00015 #include <ixwebsocket/IXUuid.h>
00016 #include <string>
00017
00018 // Versions numbers
00019 #include <ixwebsocket/IXWebSocketVersion.h>
00020 #include <yaodaq/YaodaqVersion.hpp>
00021
00022 namespace yaodaq
00023 {
00024
00025 Message::Message()
00026 {
00027   m_JSON["from"];
00028   m_JSON["to"];
00029   m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
00030   m_JSON["uuid"] = ix::uuid4();
00031   m_JSON["content"];
00032   m_JSON["timestamp"]                        = fmt::format( "{:%F %T %z}", fmt::gmtime(
       std::chrono::system_clock::to_time_t( std::chrono::system_clock::now() ) ) );
00033   m_JSON["meta"]["compiler"]                 = nlohmann::json::meta()["compiler"];
00034   m_JSON["meta"]["platform"]                 = nlohmann::json::meta()["platform"];
00035   m_JSON["meta"]["versions"]["json"]         = nlohmann::json::meta()["version"]["string"];
00036   m_JSON["meta"]["versions"]["yaodaq"]       = yaodaq_version.to_string();
00037   m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
00038 }
00039
00040 void Message::setContent( const nlohmann::json& content ) { m_JSON["content"] =
       static_cast<nlohmann::json>( content ); }
00041
00042 void Message::setContent( const std::string& content )
00043 {
00044   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00045   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00046 }
00047
00048 void Message::setContent( const char* content )
00049 {
00050   m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
00051   if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
00052 }
00053
```

```
00054 Message::Message( const nlohmann::json& content, const MessageType& messageType ) : Message(
       messageType ) { setContent( content ); }
00055
00056 Message::Message( const std::string& content, const MessageType& messageType ) : Message( messageType
       ) { setContent( content ); }
00057
00058 Message::Message( const char* content, const MessageType& messageType ) : Message( messageType ) {
       setContent( content ); }
00059
00060 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 ); }
00061
00062 nlohmann::json Message::get() const { return m_JSON; }
00063
00064 std::string Message::getTypeName() const { return m_JSON["type"].get<std::string>(); }
00065
00066 MessageType Message::getTypeValue() const { return magic_enum::enum_cast<MessageType>(
       m_JSON["type"].get<std::string>() ).value(); }
00067
00068 nlohmann::json Message::getContent() const { return m_JSON["content"]; }
00069
00070 std::string Message::getTimestamp() const { return m_JSON["timestamp"].get<std::string>(); }
00071
00072 std::time_t Message::getTime() const
00073 {
00074   std::tm tm;
00075   memset( &tm, 0, sizeof( tm ) );
00076   std::istringstream ss( getTimestamp() );
00077   ss >> std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
00078   return mktime( &tm );
00079 }
00080
00081 void Message::setFrom( const Identifier& identifier )
00082 {
00083   m_JSON["from"]["name"]   = identifier.getName();
00084   m_JSON["from"]["type"]   = identifier.getType();
00085   m_JSON["from"]["family"] = identifier.getFamily();
00086   m_JSON["from"]["class"]  = identifier.getClass();
00087   m_JSON["from"]["domain"] = identifier.getDomain();
00088 }
00089
00090 Identifier Message::getIdentifier() const
00091 {
00092   if( m_JSON["from"].is_null() ) return {};
00093   else
00094   {
00095     Identifier id( m_JSON["from"]["type"].get<std::string>(),
       m_JSON["from"]["name"].get<std::string>() );
00096     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(),
00097                     magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
       ).value() );
00098     return id;
00099   }
00100 }
00101
00102 Message::Message( const MessageType& messageType ) : Message() { m_JSON["type"] =
       magic_enum::enum_name( messageType ); }
00103
00104 // MessageException
00105 MessageException::MessageException( const Exception& exception ) : Message( MessageType::Exception )
00106 {
00107   nlohmann::json j;
00108   j["code"]          = exception.code();
00109   j["description"]   = exception.description();
00110   j["line"]          = exception.line();
00111   j["column"]        = exception.column();
00112   j["file_name"]     = exception.file_name();
00113   j["function_name"] = exception.function_name();
00114   setContent( j );
00115 }
00116
00117 std::int_least32_t MessageException::getCode() { return
       get()["content"]["code"].get<std::int_least32_t>(); }
00118
00119 std::string MessageException::getDescription() { return
       get()["content"]["description"].get<std::string>(); }
00120
00121 std::int_least32_t MessageException::getLine() { return
       get()["content"]["line"].get<std::int_least32_t>(); }
00122
00123 std::int_least32_t MessageException::getColumn() { return
       get()["content"]["column"].get<std::int_least32_t>(); }
00124
00125 std::string MessageException::getFileName() { return get()["content"]["file_name"].get<std::string>();
       }
```

```
00126
00127 std::string MessageException::getFunctionName() { return
       get()["content"]["function_name"].get<std::string>(); }
00128
00129 }  // 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/Message.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/Message.hpp"
00010 #include "yaodaq/StatusCode.hpp"
00011
00012 #include <chrono>
00013 #include <ixwebsocket/IXNetSystem.h>
00014 #include <magic_enum.hpp>
00015 #include <spdlog/sinks/stdout_color_sinks.h>
00016 #include <thread>
00017
00018 namespace yaodaq
00019 {
00020
00021 bool WebsocketClient::m_ThrowGeneralIfSameName{ true };
00022
00023 void WebsocketClient::throwGeneralIfSameName( const bool& activate ) { m_ThrowGeneralIfSameName =
      activate; }
00024
00025 WebsocketClient::WebsocketClient( const std::string& name, const std::string& type ) : m_Identifier(
      type, name )
00026 {
00027   ix::initNetSystem();
00028
00029   m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
00030   m_Logger.setName( m_Identifier.get() );
00031   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00032
00033   ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
00034   setExtraHeaders( header );
00035
00036   setOnMessageCallback(
00037     [this]( const ix::WebSocketMessagePtr& msg )
00038     {
00039       if( msg->type == ix::WebSocketMessageType::Message )
00040       {
00041         IXMessage ixmessage( msg );
00042         onMessage( ixmessage );
00043       }
00044       else if( msg->type == ix::WebSocketMessageType::Open )
00045       {
00046         Open open( msg->openInfo );
00047         onOpen( open );
00048       }
00049       else if( msg->type == ix::WebSocketMessageType::Close )
00050       {
00051         Close close( msg->closeInfo );
00052         onClose( close );
00053       }
00054       else if( msg->type == ix::WebSocketMessageType::Error )
00055       {
00056         Error error( msg->errorInfo );
00057         onError( error );
00058       }
00059       else if( msg->type == ix::WebSocketMessageType::Ping )
00060       {
00061         Ping ping( msg );
00062         onPing( ping );
00063       }
00064       else if( msg->type == ix::WebSocketMessageType::Pong )
00065       {
00066         Pong pong( msg );
00067         onPong( pong );
00068       }
00069       else if( msg->type == ix::WebSocketMessageType::Fragment )
00070       {
00071         Fragment fragment( msg );
00072         onFragment( fragment );
00073       }
00074     } );
00075 }
00076
00077 void WebsocketClient::onMessage( Message& message )
00078 {
00079   switch( message.getTypeValue() )
00080   {
00081     case MessageType::Exception:
00082       MessageException& message_exception = reinterpret_cast<MessageException&>( message );
00083       // Special case for connection to the server with the same name as an other client !
00084       if( static_cast<StatusCode>( message_exception.getCode() ) ==
      StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED )
00085       {
00086         disableAutomaticReconnection();
00087         std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
00088         close( message_exception.getCode(), message_exception.getDescription() );
00089         std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
```

```
00090         if( m_ThrowGeneralIfSameName ) throw Exception( static_cast<StatusCode>(
      message_exception.getCode() ), message_exception.getDescription() );
00091       }
00092       onException( message_exception );
00093       break;
00094     }
00095 }
00096
00097 void WebsocketClient::onOpen( Open& open )
00098 {
00099   std::string headers;
00100   for( auto it: open.getHeaders() ) { headers += fmt::format( "\t{}: {}\n", it.first, it.second ); }
00101   logger()->debug( fmt::format( fg( fmt::color::green ), "Connection opened:\nURI: {}\nProtocol:
      {}\nHeaders:\n{}", open.getURI(), open.getProtocol(), headers ) );
00102 }
00103
00104 void WebsocketClient::onClose( Close& close ) { logger()->debug( fmt::format( fg( fmt::color::green ),
      "Connection closed:\nCode: {}\nReason: {}\nRemote: {}", close.getCode(), close.getReason(),
      close.getRemote() ) ); }
00105
00106 void WebsocketClient::onError( Error& error )
00107 {
00108   logger()->error( fmt::format( fg( fmt::color::red ), "Error:\nRetries: {}\nWait time: {}\nHTTP
      status: {}\nReason: {}\nCompression error: {}", error.getRetries(), error.getWaitTime(),
      error.getHttpStatus(), error.getDecompressionError() ) );
00109 }
00110
00111 void WebsocketClient::onPing( Ping& ping ) { logger()->debug( fmt::format( fg( fmt::color::green ),
      "Ping:\n{}", ping.getContent().dump( 2 ) ) ); }
00112
00113 void WebsocketClient::onPong( Pong& pong ) { logger()->debug( fmt::format( fg( fmt::color::green ),
      "Pong:\n{}", pong.getContent().dump( 2 ) ) ); }
00114
00115 void WebsocketClient::onFragment( Fragment& fragment ) {}
00116
00117 void WebsocketClient::onException( MessageException& message )
00118 {
00119   Exception exception( StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED, message.getDescription()
      );
00120   logger()->critical( "{Exception }", exception.what() );
00121 }
00122
00123 WebsocketClient::~WebsocketClient()
00124 {
00125   stop();
00126   ix::uninitNetSystem();
00127 }
00128
00129 void WebsocketClient::start()
00130 {
00131   if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00132   {
00133     logger()->trace( "Client started. Connected to {}", getUrl() );
00134     ix::WebSocket::start();
00135   }
00136 }
00137
00138 void WebsocketClient::stop()
00139 {
00140   if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00141   {
00142     logger()->trace( "Client stopped" );
00143     ix::WebSocket::stop();
00144     while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
      std::chrono::microseconds( 1 ) ); }
00145   }
00146 }
00147
00148 void WebsocketClient::loop()
00149 {
00150   WebsocketClient::start();
00151   m_Looper.supressInstance();
00152   onRaisingSignal();
00153 }
00154
00155 void WebsocketClient::onRaisingSignal()
00156 {
00157   Signal signal = m_Looper.loop();
00158   if( m_Looper.getInstance() == 0 )
00159   {
00160     int value = magic_enum::enum_integer( signal );
00161     if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
      "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00162     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
00163     {
00164       logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00165     }
```

```
00166     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
00167     {
00168       fmt::print( "\n" );
00169       logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00170     }
00171     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
00172     {
00173       fmt::print( "\n" );
00174       logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00175     }
00176     else
00177     {
00178       fmt::print( "\n" );
00179       logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00180     }
00181     if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
      std::exit( magic_enum::enum_integer( signal ) );
00182   }
00183 }
00184
00185 }  // namespace yaodaq
```

9.59 yaodaq/WebsocketServer.cpp File Reference

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

Namespaces

- namespace yaodaq

9.60 WebsocketServer.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/WebsocketServer.hpp"
00006
00007 #include "yaodaq/Classification.hpp"
00008 #include "yaodaq/ConnectionState.hpp"
00009 #include "yaodaq/Exception.hpp"
00010 #include "yaodaq/IXWebsocketMessage.hpp"
00011 #include "yaodaq/Identifier.hpp"
00012 #include "yaodaq/StatusCode.hpp"
00013
00014 #include <chrono>
00015 #include <iostream>
00016 #include <ixwebsocket/IXNetSystem.h>
00017 #include <magic_enum.hpp>
00018 #include <spdlog/sinks/stdout_color_sinks.h>
00019 #include <spdlog/spdlog.h>
00020 #include <string>
00021 #include <thread>
00022 #include <utility>
00023
00024 namespace yaodaq
00025 {
```

```
00026
00027 WebsocketServer::WebsocketServer( const std::string& name, const int& port, const std::string& host,
      const int& backlog, const std::size_t& maxConnections, const int& handshakeTimeoutSecs, const int&
      addressFamily, const std::string& type ) :
00028   ix::WebSocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
      m_Identifier( type, name )
00029 {
00030   ix::initNetSystem();
00031
00032   m_Identifier.generateKey( Domain::Application, Class::Server, Family::WebSocketServer );
00033   m_Logger.setName( m_Identifier.get() );
00034   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00035
00036   setConnectionStateFactory( []() { return std::make_shared<ConnectionState>(); } );
00037
00038   setOnClientMessageCallback(
00039     [this]( std::shared_ptr<ix::ConnectionState> connectionState, ix::WebSocket& webSocket, const
      ix::WebSocketMessagePtr& msg )
00040     {
00041       // The ConnectionState object contains information about the connection
00042       std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>(
      connectionState );
00043       if( msg->type == ix::WebSocketMessageType::Message )
00044       {
00045         IXMessage ixmessage( msg );
00046         // sendToLoggers(ixmessage,webSocket);
00047         onMessage( ixmessage );
00048       }
00049       else if( msg->type == ix::WebSocketMessageType::Open )
00050       {
00051         // Check if a client with the same name is already connected;
00052         connection->computeId( getHost() + ":" + std::to_string( getPort() ), Identifier::parse(
      msg->openInfo.headers["id"] ) );
00053         if( connection->isTerminated() )
00054         {
00055           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() )
      );
00056           MessageException message_exception(
00057             Exception( StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED, fmt::format( "One client
      with the name \"{}\" is already connected to ws{}://{}:{} !", Identifier::parse(
      msg->openInfo.headers["id"] ).getName(), "", getHost(), getPort() ) ) );
00058
00059           // Send to the client on exception;
00060           webSocket.send( message_exception.dump() );
00061           // Send to loggers except the client as it has been send before;
00062           // sendToLoggers(message_exception,webSocket);
00063
00064           webSocket.stop( magic_enum::enum_integer(
      StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED ),
00065                           fmt::format( "One client with the name \"{}\" is already connected to
      ws{}://{}:{} !", Identifier::parse( msg->openInfo.headers["id"] ).getName(), "", getHost(), getPort()
      ) );
00066           std::this_thread::sleep_for( std::chrono::milliseconds( 50 ) );
00067           return;
00068         }
00069         addClient( Identifier::parse( msg->openInfo.headers["id"] ), webSocket );
00070         Open open( msg->openInfo, connection );
00071         // sendToLoggers( open, webSocket );
00072         onOpen( open );
00073       }
00074       else if( msg->type == ix::WebSocketMessageType::Close )
00075       {
00076         Close close( msg->closeInfo, connection );
00077         // sendToLoggers( close, webSocket );
00078         onClose( close );
00079         removeClient( webSocket );
00080       }
00081       else if( msg->type == ix::WebSocketMessageType::Error )
00082       {
00083         Error error( msg->errorInfo, connection );
00084         // sendToLoggers( error, webSocket );
00085         onError( error );
00086       }
00087       else if( msg->type == ix::WebSocketMessageType::Ping )
00088       {
00089         Ping ping( msg, connection );
00090         // sendToLoggers( ping, webSocket );
00091         onPing( ping );
00092       }
00093       else if( msg->type == ix::WebSocketMessageType::Pong )
00094       {
00095         Pong pong( msg, connection );
00096         // sendToLoggers( pong, webSocket );
00097         onPong( pong );
00098       }
00099       else if( msg->type == ix::WebSocketMessageType::Fragment )
```

```
00100       {
00101         Fragment fragment( msg, connection );
00102         // sendToLoggers( fragment, webSocket );
00103         onFragment( fragment );
00104       }
00105     } );
00106 }
00107
00108 void WebsocketServer::addClient( const Identifier& identifier, ix::WebSocket& websocket )
00109 {
00110   std::lock_guard<std::mutex> guard( m_Mutex );
00111   m_Clients.try_emplace( identifier, websocket );
00112 }
00113
00114 void WebsocketServer::removeClient( ix::WebSocket& websocket )
00115 {
00116   std::lock_guard<std::mutex> guard( m_Mutex );
00117   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00118   {
00119     if( &it->second == &websocket )
00120     {
00121       m_Clients.erase( it->first );
00122       break;
00123     }
00124   }
00125 }
00126
00127 void WebsocketServer::sendToLoggers( Message& message, ix::WebSocket& webSocket )
00128 {
00129   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00130   {
00131     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
      != &it->second ) it->second.send( message.dump() );
00132   }
00133 }
00134
00135 void WebsocketServer::sendToLoggers( Message& message )
00136 {
00137   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00138   {
00139     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
      it->second.send( message.dump() );
00140   }
00141 }
00142
00143 void WebsocketServer::sendToLoggers( const Message& message, ix::WebSocket& webSocket )
00144 {
00145   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00146   {
00147     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket
      != &it->second ) it->second.send( message.dump() );
00148   }
00149 }
00150
00151 void WebsocketServer::sendToLoggers( const Message& message )
00152 {
00153   for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end();
      ++it )
00154   {
00155     if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger )
      it->second.send( message.dump() );
00156   }
00157 }
00158
00159 void WebsocketServer::onMessage( Message& message )
00160 {
00161   switch( message.getTypeValue() )
00162   {
00163     case MessageType::Exception:
00164     {
00165       MessageException& message_exception = reinterpret_cast<MessageException&>( message );
00166       onException( message_exception );
00167     }
00168     default:
00169     {
00170       onUnknown( message );
00171     }
00172   }
00173 }
00174
00175 void WebsocketServer::onOpen( Open& open )
00176 {
00177   std::string headers;
```

```
00178   for( auto it: open.getHeaders() ) { headers += fmt::format( "\t{}: {}\n", it.first, it.second ); }
00179   logger()->debug( fmt::format( fg( fmt::color::green ), "Connection opened:\nId: {}\nRemote IP:
       {}\nRemote port: {}\nURI: {}\nProtocol: {}\nHeaders:\n{}", open.getId(), open.getRemoteIp(),
       open.getRemotePort(), open.getURI(), open.getProtocol(), headers ) );
00180 }
00181
00182 void WebsocketServer::onClose( Close& close )
00183 {
00184   logger()->debug(
00185     fmt::format( fg( fmt::color::green ), "Connection closed:\nId: {}\nRemote IP: {}\nRemote port:
       {}\nCode: {}\nReason: {}\nRemote: {}", close.getId(), close.getRemoteIp(), close.getRemotePort(),
       close.getCode(), close.getReason(), close.getRemote() ) );
00186 }
00187
00188 void WebsocketServer::onError( Error& error )
00189 {
00190   logger()->error( fmt::format( fg( fmt::color::red ), "Error:\nId: {}\nRemote IP: {}\nRemote port:
       {}\nRetries: {}\nWait time: {}\nHTTP status: {}\nReason: {}\nCompression error: {}", error.getId(),
       error.getRemoteIp(), error.getRemotePort(),
00191                                 error.getRetries(), error.getWaitTime(), error.getHttpStatus(),
       error.getDecompressionError() ) );
00192 }
00193
00194 void WebsocketServer::onPing( Ping& ping ) { logger()->debug( fmt::format( fg( fmt::color::green ),
       "Ping:\n{}", ping.getContent().dump( 2 ) ) ); }
00195
00196 void WebsocketServer::onPong( Pong& pong ) { logger()->debug( fmt::format( fg( fmt::color::green ),
       "Pong:\n{}", pong.getContent().dump( 2 ) ) ); }
00197
00198 void WebsocketServer::onFragment( Fragment& fragment ) {}
00199
00200 void WebsocketServer::onException( MessageException& message ) {}
00201
00202 void WebsocketServer::onUnknown(Message& unknown)
00203 {
00204   logger()->error( fmt::format( fg( fmt::color::red ), "Unknwown:\n{}", unknown.dump( 2 ) ) );
00205 }
00206
00207 void WebsocketServer::listen()
00208 {
00209   if( !m_isListening )
00210   {
00211     std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
00212     if( ret.first )
00213     {
00214       m_isListening = ret.first;
00215       logger()->info( "Server listening on {0}:{1}", getHost(), getPort() );
00216     }
00217     else
00218       throw Exception( StatusCode::LISTEN_ERROR, ret.second );
00219   }
00220 }
00221
00222 void WebsocketServer::start()
00223 {
00224   if( !m_isStarted )
00225   {
00226     m_isStarted = true;
00227     logger()->trace( "Server started" );
00228     ix::WebSocketServer::start();
00229   }
00230 }
00231
00232 void WebsocketServer::stop( bool useless )
00233 {
00234   if( !m_isStopped )
00235   {
00236     m_isStopped = true;
00237     useless     = !useless;
00238     logger()->trace( "Server stopped" );
00239     ix::WebSocketServer::stop();
00240   }
00241 }
00242
00243 void WebsocketServer::setVerbosity( const yaodaq::LoggerHandler::Verbosity& verbosity ) {
       m_Logger.setVerbosity( verbosity ); }
00244
00245 WebsocketServer::~WebsocketServer()
00246 {
00247   stop();
00248   ix::uninitNetSystem();
00249 }
00250
00251 void WebsocketServer::loop()
00252 {
00253   listen();
00254   start();
```

```
00255   m_Looper.supressInstance();
00256   onRaisingSignal();
00257 }
00258
00259 void WebsocketServer::onRaisingSignal()
00260 {
00261   Signal signal = m_Looper.loop();
00262   if( m_Looper.getInstance() == 0 )
00263   {
00264     int value = magic_enum::enum_integer( signal );
00265     if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
       "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00266     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
00267     {
00268       logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00269     }
00270     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
00271     {
00272       fmt::print( "\n" );
00273       logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00274     }
00275     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
00276     {
00277       fmt::print( "\n" );
00278       logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00279     }
00280     else
00281     {
00282       fmt::print( "\n" );
00283       logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00284     }
00285     if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
       std::exit( magic_enum::enum_integer( signal ) );
00286   }
00287 }
00288
00289 }  // namespace yaodaq
```