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:

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

The following software may be included in this product: zlib-ng. 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

2.1.2 Original SSLeay License

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

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

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

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

The following software may be included in this product: CLI11. This software contains the following license and notice below:

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

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.

```
{
  Unknown = 0,
  Server,
  Client,
  // Module is a client with start stop etc...
  Module,
  // Board is a module with a connector
  Board,
};
```

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.

```
{
  Unknown     = 0,
  Application = 1,
  Web         = 2,
};
```

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.

```
{
  Unknown = 0,
  WebSocketServer,
  WebSocketClient,
  Logger,
  Controller,
  Configurator,
  SlowController,
  Viewer,
  Analyser,
  FileWriter,
};
```

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.

```
{
  // IXWebSocket MessageType (Message is not set here)
  Open = -6,
  Close,
  Error,
  Ping,
  Pong,
  Fragment,
  // Unknown should not be used !
  Unknown = 0,
  Exception,
};
```

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.

```
{
  Info     = 1,
  Warning  = 10,
  Error    = 100,
  Critical = 1000,
};
```

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.

```
{
  NO   = 0,  // No Signal.
  // Critical
  ABRT = static_cast<int>( Severity::Critical ) + 1,  // (Signal Abort) Abnormal termination, such as
 is initiated by the abort function.
  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).
  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.
  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.
  // Warning
  INT  = static_cast<int>( Severity::Warning ) + 1,  // (Signal Interrupt) Interactive attention
signal. Generally generated by the application user.
  TERM = static_cast<int>( Severity::Warning ) + 2,  // (Signal Terminate) Termination request sent to
program.
};
```

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.

```
{
  SUCCESS = 0,
  LISTEN_ERROR,
  WRONG_NUMBER_PARAMETERS,
  CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED = 4999,
};
```

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.

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

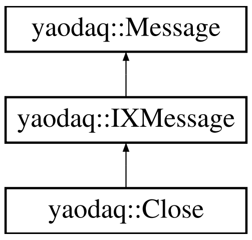
Chapter 8

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.

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

8.1.3.19 setFrom()

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

Definition at line 81 of file Message.cpp.

```
{
  m_JSON["from"]["name"]   = identifier.getName();
  m_JSON["from"]["type"]   = identifier.getType();
  m_JSON["from"]["family"] = identifier.getFamily();
  m_JSON["from"]["class"]  = identifier.getClass();
  m_JSON["from"]["domain"] = identifier.getDomain();
}
```

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.

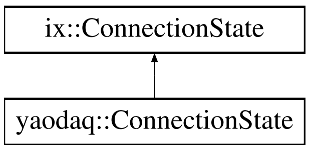
The documentation for this class was generated from the following files:

- 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 }
```

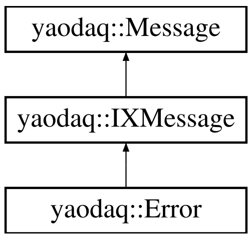
The documentation for this class was generated from the following files:

- 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.

```
                                                  : IXMessage( MessageType::Error )
{
  nlohmann::json j;
  j["retries"]            = errorInfo.retries;
  j["wait_time"]          = errorInfo.wait_time;
  j["http_status"]        = errorInfo.http_status;
  j["reason"]             = errorInfo.reason;
  j["decompression_error"] = errorInfo.decompressionError;
  setContent( j );
}
```

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.

```
{ 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.

```
{ 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.

```
{ 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.

```
{
  nlohmann::json j = getContent();
  j["id"]          = connectionState->getId();
  j["remote_ip"]   = connectionState->getRemoteIp();
  j["remote_port"] = connectionState->getRemotePort();
  setContent( j );
}
```

8.3.3.18 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

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

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.

```
{ 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.

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

8.3.3.21 setFrom()

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

Definition at line 81 of file Message.cpp.

```
{
  m_JSON["from"]["name"]   = identifier.getName();
  m_JSON["from"]["type"]   = identifier.getType();
  m_JSON["from"]["family"] = identifier.getFamily();
  m_JSON["from"]["class"]  = identifier.getClass();
  m_JSON["from"]["domain"] = identifier.getDomain();
}
```

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.

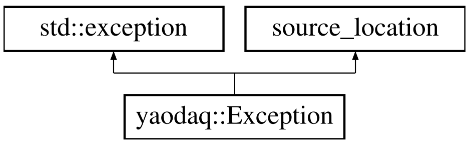
The documentation for this class was generated from the following files:

- 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(); }
```

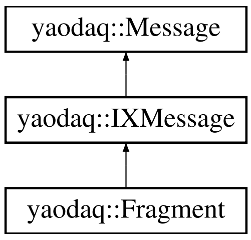
The documentation for this class was generated from the following files:

- 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.

```
{ 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.

```
{ 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.

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

8.5.3.9 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
{ 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.

```
{ 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.

```
{ 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.

```
{
  nlohmann::json j = getContent();
  j["id"]          = connectionState->getId();
  j["remote_ip"]   = connectionState->getRemoteIp();
  j["remote_port"] = connectionState->getRemotePort();
  setContent( j );
}
```

8.5.3.13 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

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

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.

```
{ 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.

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

8.5.3.16 setFrom()

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

Definition at line 81 of file Message.cpp.

```
{
  m_JSON["from"]["name"]   = identifier.getName();
  m_JSON["from"]["type"]   = identifier.getType();
  m_JSON["from"]["family"] = identifier.getFamily();
  m_JSON["from"]["class"]  = identifier.getClass();
  m_JSON["from"]["domain"] = identifier.getDomain();
}
```

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.

```
{
  std::vector<std::string> result;
  std::string              tmp        = id;
  std::string              separator  = "/";
  std::size_t              second_pos = tmp.find( separator );
  while( second_pos != std::string::npos )
  {
    if( 0 != second_pos )
    {
      std::string word = tmp.substr( 0, second_pos - 0 );
      result.push_back( word );
    }
    else
      result.push_back( "" );
    tmp        = tmp.substr( second_pos + separator.length() );
    second_pos = tmp.find( separator );
    if( second_pos == std::string::npos ) result.push_back( tmp );
  }
  if( result.size() == 5 )
  {
    Identifier identifier( result[3], result[4] );
    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()
);
    return identifier;
  }
  else
  {
    throw Exception( StatusCode::WRONG_NUMBER_PARAMETERS, "Number of parameters in key should be 5
(Domain/Class/Family/Type/Name) !" );
  }
}
```

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.

```
{ 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 }
```

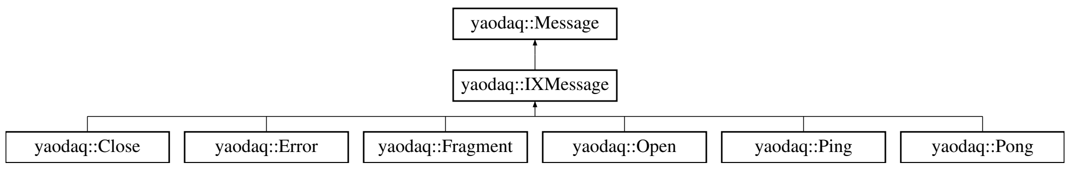
The documentation for this class was generated from the following files:

- 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.

```
{ 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.

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

8.8.3.9 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
{ 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.

```
{ 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.

```
{ 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.

```
{
  nlohmann::json j = getContent();
  j["id"]          = connectionState->getId();
  j["remote_ip"]   = connectionState->getRemoteIp();
  j["remote_port"] = connectionState->getRemotePort();
  setContent( j );
}
```

8.8.3.13 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

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

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.

```
{ 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.

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

8.8.3.16 setFrom()

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

Definition at line 81 of file Message.cpp.

```
{
  m_JSON["from"]["name"]   = identifier.getName();
  m_JSON["from"]["type"]   = identifier.getType();
  m_JSON["from"]["family"] = identifier.getFamily();
  m_JSON["from"]["class"]  = identifier.getClass();
  m_JSON["from"]["domain"] = identifier.getDomain();
}
```

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.

```
{
  Off,
  Trace,
  Debug,
  Info,
  Warn,
  Error,
  Critical
};
```

8.10.3 Constructor & Destructor Documentation

8.10.3.1 LoggerHandler()

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

Definition at line 12 of file LoggerHandler.cpp.

```
{ 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.

```
{
  if( m_hasBeenAdded == false )
  {
    m_hasBeenAdded = true;
    ++m_instance;
  }
}
```

8.11.2.2 ∼Looper()

`yaodaq::Looper::∼Looper ( )`

Definition at line 52 of file Looper.cpp.

```
{
  if( m_hasBeenAdded == true && m_hasBeenSupressed == false )
  {
    m_hasBeenSupressed = true;
    --m_instance;
  }
}
```

8.11.3 Member Function Documentation

8.11.3.1 getInstance()

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

Definition at line 17 of file Looper.cpp.

```
{ return m_instance; }
```

8.11.3.2 getSignal()

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

Definition at line 50 of file Looper.cpp.

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

8.11.3.3 loop()

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

Definition at line 37 of file Looper.cpp.

```
{
  static Signal signal{ yaodaq::Signal::NO };
  if( m_instance == 0 )
  {
    do {
      signal = m_Interrupt.getSignal();
      std::this_thread::sleep_for( std::chrono::microseconds( 1 ) );
    } while( signal == yaodaq::Signal::NO );
  }
  return signal;
}
```

8.11.3.4 supressInstance()

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

Definition at line 19 of file Looper.cpp.

```
{
  if( m_hasBeenSupressed == false )
  {
    m_hasBeenSupressed = true;
    m_instance--;
  }
}
```

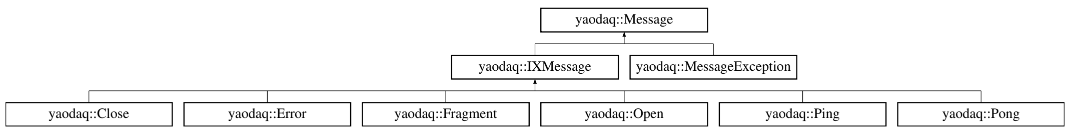
The documentation for this class was generated from the following files:

- 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.

```
{
  m_JSON["from"];
  m_JSON["to"];
  m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
  m_JSON["uuid"] = ix::uuid4();
  m_JSON["content"];
  m_JSON["timestamp"]                        = fmt::format( "{:%F %T %z}", fmt::gmtime(
 std::chrono::system_clock::to_time_t( std::chrono::system_clock::now() ) ) );
  m_JSON["meta"]["compiler"]                 = nlohmann::json::meta()["compiler"];
  m_JSON["meta"]["platform"]                 = nlohmann::json::meta()["platform"];
  m_JSON["meta"]["versions"]["json"]         = nlohmann::json::meta()["version"]["string"];
  m_JSON["meta"]["versions"]["yaodaq"]       = yaodaq_version.to_string();
  m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
}
```

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.

```
: 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.

```
: 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.

```
: 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.

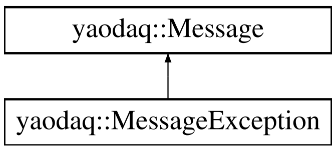
The documentation for this class was generated from the following files:

- 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.

```
                                                          : Message( MessageType::Exception )
{
  nlohmann::json j;
  j["code"]          = exception.code();
  j["description"]   = exception.description();
  j["line"]          = exception.line();
  j["column"]        = exception.column();
  j["file_name"]     = exception.file_name();
  j["function_name"] = exception.function_name();
  setContent( j );
}
```

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.

```
{ 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.

```
{ return m_JSON; }
```

8.13.3.3 getCode()

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

Definition at line 117 of file Message.cpp.

```
{ 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.

```
{ 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.

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

8.13.3.6 getDescription()

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

Definition at line 119 of file Message.cpp.

```
{ 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.

```
{ 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.

```
{ 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.

```
{
  if( m_JSON["from"].is_null() ) return {};
  else
  {
    Identifier id( m_JSON["from"]["type"].get<std::string>(),
  m_JSON["from"]["name"].get<std::string>() );
    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(),
                    magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>()
  ).value() );
    return id;
  }
}
```

8.13.3.10 getLine()

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

Definition at line 121 of file Message.cpp.

```
{ 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.

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

8.13.3.12 getTimestamp()

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

Definition at line 70 of file Message.cpp.

```
{ 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.

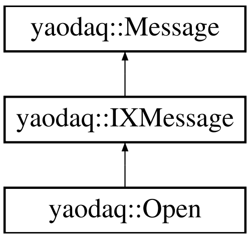
The documentation for this class was generated from the following files:

- 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.

```
                                                 : IXMessage( MessageType::Open )
{
  nlohmann::json j = getContent();
  j["uri"]         = openInfo.uri;
  j["headers"]     = openInfo.headers;
  j["protocol"]    = openInfo.protocol;
  setContent( j );
}
```

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.

```
: 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.

```
{ 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.

```
{ return m_JSON; }
```

8.14.3.3 getContent()

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

Definition at line 68 of file Message.cpp.

```
{ 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.

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

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.

```
{
  m_JSON["from"]["name"]   = identifier.getName();
  m_JSON["from"]["type"]   = identifier.getType();
  m_JSON["from"]["family"] = identifier.getFamily();
  m_JSON["from"]["class"]  = identifier.getClass();
  m_JSON["from"]["domain"] = identifier.getDomain();
}
```

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.

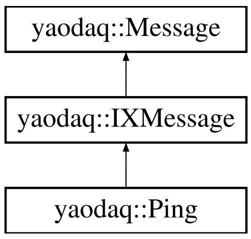
The documentation for this class was generated from the following files:

- 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

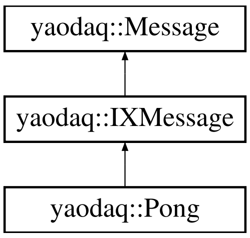
8.15.4.1 m_JSON

`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.

```
{ 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.

```
{
  nlohmann::json j = getContent();
  j["id"]          = connectionState->getId();
  j["remote_ip"]   = connectionState->getRemoteIp();
  j["remote_port"] = connectionState->getRemotePort();
  setContent( j );
}
```

8.16.3.13 setContent() [1/3]

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

Definition at line 48 of file Message.cpp.

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

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.

```
{ 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.

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

8.16.3.16 setFrom()

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

Definition at line 81 of file Message.cpp.

```
{
  m_JSON["from"]["name"]   = identifier.getName();
  m_JSON["from"]["type"]   = identifier.getType();
  m_JSON["from"]["family"] = identifier.getFamily();
  m_JSON["from"]["class"]  = identifier.getClass();
  m_JSON["from"]["domain"] = identifier.getDomain();
}
```

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.

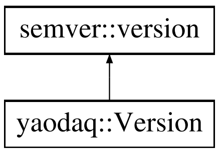
The documentation for this class was generated from the following files:

- 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; }
```

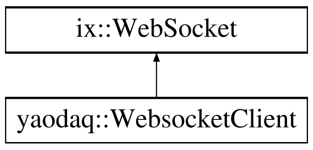
The documentation for this class was generated from the following files:

- 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.

```
                                                                               : m_Identifier( type,
       name )
{
  ix::initNetSystem();

  m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
  m_Logger.setName( m_Identifier.get() );
  m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );

  ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
  setExtraHeaders( header );

  setOnMessageCallback(
    [this]( const ix::WebSocketMessagePtr& msg )
    {
      if( msg->type == ix::WebSocketMessageType::Message )
      {
        IXMessage ixmessage( msg );
        onMessage( ixmessage );
      }
      else if( msg->type == ix::WebSocketMessageType::Open )
      {
        Open open( msg->openInfo );
        onOpen( open );
      }
      else if( msg->type == ix::WebSocketMessageType::Close )
      {
        Close close( msg->closeInfo );
        onClose( close );
      }
      else if( msg->type == ix::WebSocketMessageType::Error )
      {
        Error error( msg->errorInfo );
        onError( error );
      }
      else if( msg->type == ix::WebSocketMessageType::Ping )
      {
        Ping ping( msg );
        onPing( ping );
      }
      else if( msg->type == ix::WebSocketMessageType::Pong )
      {
        Pong pong( msg );
        onPong( pong );
      }
      else if( msg->type == ix::WebSocketMessageType::Fragment )
      {
        Fragment fragment( msg );
        onFragment( fragment );
      }
    } );
}
```

8.18.2.2 ∼WebsocketClient()

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

Definition at line 123 of file WebsocketClient.cpp.

```
{
  stop();
  ix::uninitNetSystem();
}
```

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.

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

8.18.3.2 loop()

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

Definition at line 148 of file WebsocketClient.cpp.

```
{
  WebsocketClient::start();
  m_Looper.supressInstance();
  onRaisingSignal();
}
```

8.18.3.3 onClose()

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

Definition at line 104 of file WebsocketClient.cpp.

```
{ 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.

```
{
  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() ) );
}
```

8.18.3.5 onException()

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

Definition at line 117 of file WebsocketClient.cpp.

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

8.18.3.6 onFragment()

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

Definition at line 115 of file WebsocketClient.cpp.

```
{}
```

8.18.3.7 onMessage()

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

Definition at line 77 of file WebsocketClient.cpp.

```
{
  switch( message.getTypeValue() )
  {
    case MessageType::Exception:
      MessageException& message_exception = reinterpret_cast<MessageException&>( message );
      // Special case for connection to the server with the same name as an other client !
      if( static_cast<StatusCode>( message_exception.getCode() ) ==
StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED )
      {
        disableAutomaticReconnection();
        std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
        close( message_exception.getCode(), message_exception.getDescription() );
        std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
        if( m_ThrowGeneralIfSameName ) throw Exception( static_cast<StatusCode>(
message_exception.getCode() ), message_exception.getDescription() );
      }
      onException( message_exception );
      break;
  }
}
```

8.18.3.8 onOpen()

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

Definition at line 97 of file WebsocketClient.cpp.

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

8.18.3.9 onPing()

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

Definition at line 111 of file WebsocketClient.cpp.

```
{ 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.

```
{ 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.

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

8.18.3.12 stop()

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

Definition at line 138 of file WebsocketClient.cpp.

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

8.18.3.13 throwGeneralIfSameName()

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

Definition at line 23 of file WebsocketClient.cpp.

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

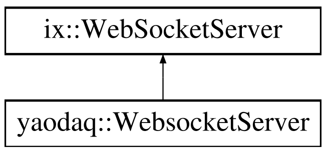
The documentation for this class was generated from the following files:

- 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
```

```
      std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>( connectionState );
      if( msg->type == ix::WebSocketMessageType::Message )
      {
        IXMessage ixmessage( msg );
        // sendToLoggers(ixmessage,webSocket);
        onMessage( ixmessage );
      }
      else if( msg->type == ix::WebSocketMessageType::Open )
      {
        // Check if a client with the same name is already connected;
        connection->computeId( getHost() + ":" + std::to_string( getPort() ), Identifier::parse( msg->openInfo.headers["id"] ) );
        if( connection->isTerminated() )
        {
          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() ) );
          MessageException message_exception(
            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() ) ) );

          // Send to the client on exception;
          webSocket.send( message_exception.dump() );
          // Send to loggers except the client as it has been send before;
          // sendToLoggers(message_exception,webSocket);

          webSocket.stop( magic_enum::enum_integer( 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() ) );
          std::this_thread::sleep_for( std::chrono::milliseconds( 50 ) );
          return;
        }
        addClient( Identifier::parse( msg->openInfo.headers["id"] ), webSocket );
        Open open( msg->openInfo, connection );
        // sendToLoggers( open, webSocket );
        onOpen( open );
      }
      else if( msg->type == ix::WebSocketMessageType::Close )
      {
        Close close( msg->closeInfo, connection );
        // sendToLoggers( close, webSocket );
        onClose( close );
        removeClient( webSocket );
      }
      else if( msg->type == ix::WebSocketMessageType::Error )
      {
        Error error( msg->errorInfo, connection );
        // sendToLoggers( error, webSocket );
        onError( error );
      }
      else if( msg->type == ix::WebSocketMessageType::Ping )
      {
        Ping ping( msg, connection );
        // sendToLoggers( ping, webSocket );
        onPing( ping );
      }
      else if( msg->type == ix::WebSocketMessageType::Pong )
      {
        Pong pong( msg, connection );
        // sendToLoggers( pong, webSocket );
        onPong( pong );
      }
      else if( msg->type == ix::WebSocketMessageType::Fragment )
      {
        Fragment fragment( msg, connection );
        // sendToLoggers( fragment, webSocket );
        onFragment( fragment );
      }
    } );
}
```

8.19.2.2 ∼WebsocketServer()

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

Definition at line 245 of file WebsocketServer.cpp.

```
{
  stop();
  ix::uninitNetSystem();
}
```

8.19.3 Member Function Documentation

8.19.3.1 listen()

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

Definition at line 207 of file WebsocketServer.cpp.

```
{
  if( !m_isListening )
  {
    std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
    if( ret.first )
    {
      m_isListening = ret.first;
      logger()->info( "Server listening on {0}:{1}", getHost(), getPort() );
    }
    else
      throw Exception( StatusCode::LISTEN_ERROR, ret.second );
  }
}
```

8.19.3.2 logger()

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

Definition at line 57 of file WebsocketServer.hpp.

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

8.19.3.3 loop()

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

Definition at line 251 of file WebsocketServer.cpp.

```
{
  listen();
  start();
  m_Looper.supressInstance();
  onRaisingSignal();
}
```

8.19.3.4 onClose()

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

Definition at line 182 of file WebsocketServer.cpp.

```
{
  logger()->debug(
    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() ) );
}
```

8.19.3.5 onError()

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

Definition at line 188 of file WebsocketServer.cpp.

```
{
  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(),
                              error.getRetries(), error.getWaitTime(), error.getHttpStatus(),
error.getDecompressionError() ) );
}
```

8.19.3.6 onException()

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

Definition at line 200 of file WebsocketServer.cpp.

```
{}
```

8.19.3.7 onFragment()

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

Definition at line 198 of file WebsocketServer.cpp.

```
{}
```

8.19.3.8 onMessage()

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

Definition at line 159 of file WebsocketServer.cpp.

```
{
  switch( message.getTypeValue() )
  {
    case MessageType::Exception:
    {
      MessageException& message_exception = reinterpret_cast<MessageException&>( message );
      onException( message_exception );
    }
    default:
    {
      onUnknown( message );
    }
  }
}
```

8.19.3.9 onOpen()

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

Definition at line 175 of file WebsocketServer.cpp.

```
{
  std::string headers;
  for( auto it: open.getHeaders() ) { headers += fmt::format( "\t{}: {}\n", it.first, it.second ); }
  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 ) );
}
```

8.19.3.10 onPing()

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

Definition at line 194 of file WebsocketServer.cpp.

```
{ 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.

```
{ 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.

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

8.19.3.13 sendToLoggers() [1/4]

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

Definition at line 151 of file WebsocketServer.cpp.

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

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.

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

8.19.3.15 sendToLoggers() [3/4]

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

Definition at line 135 of file WebsocketServer.cpp.

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

8.19.3.16 sendToLoggers() [4/4]

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

Definition at line 127 of file WebsocketServer.cpp.

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

8.19.3.17 setVerbosity()

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

Definition at line 243 of file WebsocketServer.cpp.

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

8.19.3.18 start()

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

Definition at line 222 of file WebsocketServer.cpp.

```
{
  if( !m_isStarted )
  {
    m_isStarted = true;
    logger()->trace( "Server started" );
    ix::WebSocketServer::start();
  }
}
```

8.19.3.19 stop()

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

Definition at line 232 of file WebsocketServer.cpp.

```
{
  if( !m_isStopped )
  {
    m_isStopped = true;
    useless     = !useless;
    logger()->trace( "Server stopped" );
    ix::WebSocketServer::stop();
  }
}
```

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.

```
#ifndef YAODAQ_CLASSIFICATION
#define YAODAQ_CLASSIFICATION

#include <cstdint>

namespace yaodaq
{

/* The domain specify if we are on browser or standalone program */
enum class Domain : std::uint_least8_t
{
  Unknown     = 0,
  Application = 1,
  Web         = 2,
};

/* The class define if we are a server, module, or board */
enum class Class : std::uint_least8_t
{
  Unknown = 0,
  Server,
  Client,
  // Module is a client with start stop etc...
  Module,
  // Board is a module with a connector
  Board,
};

/* the family */
enum class Family : std::uint_least16_t
{
  Unknown = 0,
  WebSocketServer,
  WebSocketClient,
  Logger,
  Controller,
  Configurator,
  SlowController,
  Viewer,
  Analyser,
  FileWriter,
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_CONNECTIONSTATE
#define YAODAQ_CONNECTIONSTATE

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

namespace yaodaq
{

class Identifier;

class ConnectionState : public ix::ConnectionState
{
public:
  virtual void computeId( const std::string& host, const Identifier& id ) final;
  ConnectionState();
  virtual ~ConnectionState();

private:
  static std::list<std::pair<std::string, std::string>> m_Ids;
  std::pair<std::string, std::string>                   m_Pair;
  std::mutex                                            m_Mutex;
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_EXCEPTION
#define YAODAQ_EXCEPTION

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

namespace yaodaq
{

enum class StatusCode : std::int_least32_t;

class Exception : public std::exception, public source_location
{
public:
  Exception() = delete;

  static void setFormat( const std::string& format ) { m_Format = format; }

  static void setStyle( const fmt::text_style& style = {} ) { m_Style = style; }

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

private:
  static fmt::text_style   m_Style;
  static std::string       m_Format;
  const std::int_least32_t m_Code{ 0 };
  std::string              m_Description;
  std::string              m_Message;
  void                     constructMessage();
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_IDENTIFIER
#define YAODAQ_IDENTIFIER

#include "yaodaq/Key.hpp"

#include <cstdint>
#include <string>

namespace yaodaq
{

class Identifier
{
public:
  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 );
  [[nodiscard]] std::string getDomain() const;
  [[nodiscard]] std::string getClass() const;
  [[nodiscard]] std::string getFamily() const;
  [[nodiscard]] std::string getType() const;
  [[nodiscard]] std::string getName() const;
  [[nodiscard]] Key         getKey() const;
  [[nodiscard]] std::string get() const;
  bool                       empty() const;
  static Identifier          parse( const std::string& );
  bool                       operator<( const Identifier& ) const;

private:
  std::string m_Type{ "Unknown" };
  std::string m_Name{ "Unknown" };
  Key         m_Key;
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_HANDLER
#define YAODAQ_HANDLER

#include "yaodaq/Signal.hpp"

#include <atomic>
#include <csignal>
#include <mutex>

namespace yaodaq
{

enum class Signal;

class Interrupt
{
public:
  Interrupt();
  void   init();
  void   restore();
  Signal getSignal();
  ~Interrupt();

private:
  volatile static std::atomic<Signal> m_Signal;
  void                                setSignal( const Signal& signal );
  std::mutex                          m_mutex;
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_IXWEBSOCKETMESSAGE
#define YAODAQ_IXWEBSOCKETMESSAGE

#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>

namespace yaodaq
{

class IXMessage : public Message
{
public:
  explicit IXMessage( const MessageType& messageType );
  explicit IXMessage( const ix::WebSocketMessagePtr& msg );
  std::string getId() const;
  std::string getRemoteIp() const;
  int getRemotePort() const;
protected:
  void setConnectionStateInfos( std::shared_ptr<ConnectionState>& connectionState );
};

class Open : public IXMessage
{
public:
  explicit 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;
};

class Close : public IXMessage
{
public:
  explicit 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;
};

class Error : public IXMessage
{
public:
  explicit 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;
};

class Ping : public IXMessage
{
public:
  explicit Ping( const ix::WebSocketMessagePtr& ping );
  Ping( const ix::WebSocketMessagePtr& ping, std::shared_ptr<ConnectionState>& connectionState );
};

class Pong : public IXMessage
{
public:
  explicit Pong( const ix::WebSocketMessagePtr& pong );
  Pong( const ix::WebSocketMessagePtr& pong, std::shared_ptr<ConnectionState>& connectionState );
};

class Fragment : public IXMessage
{
public:
  explicit Fragment( const ix::WebSocketMessagePtr& fragment );
  Fragment( const ix::WebSocketMessagePtr& fragment, std::shared_ptr<ConnectionState>& connectionState
);
};

}  // namespace yaodaq
#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.

```
#ifndef YAODAQ_KEY
#define YAODAQ_KEY

#include "yaodaq/Classification.hpp"

#include <cstdint>

namespace yaodaq
{

class Key
{
private:
  std::int_least32_t m_Key{ 0 };

public:
  Key() = default;
  explicit Key( const Domain& domain, const Class& c_lass, const Family& family );
  [[nodiscard]] std::int_least8_t  getDomain() const;
  [[nodiscard]] std::int_least8_t  getClass() const;
  [[nodiscard]] std::int_least16_t getFamily() const;
  [[nodiscard]] std::int_least32_t getKey() const;
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_LOGGERHANDLER
#define YAODAQ_LOGGERHANDLER

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

namespace spdlog
{
using sink_ptr = std::shared_ptr<spdlog::sinks::sink>;
}

namespace yaodaq
{

class LoggerHandler
{
public:
  enum class Verbosity
  {
    Off,
    Trace,
    Debug,
    Info,
    Warn,
    Error,
    Critical
  };
  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();

private:
  std::shared_ptr<spdlog::logger> m_Logger{ nullptr };
  std::vector<spdlog::sink_ptr>   m_Sinks;
  std::string                     m_Name{ "Unknown" };
  Verbosity                       m_Verbosity{ Verbosity::Trace };
  void                            init();
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_LOOPER
```

```
#define YAODAQ_LOOPER

#include "yaodaq/Interrupt.hpp"

namespace yaodaq
{

enum class Signal;

class Looper
{
public:
  Looper();
  Signal      loop();
  Signal      getSignal();
  static int getInstance();
  void        supressInstance();
  ~Looper();

private:
  static int        m_instance;
  bool              m_hasBeenAdded{ false };
  bool              m_hasBeenSupressed{ false };
  static Interrupt m_Interrupt;
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_MESSAGE
#define YAODAQ_MESSAGE

#include "nlohmann/json.hpp"
#include "yaodaq/MessageType.hpp"

#include <string>

namespace yaodaq
{

class Identifier;
class Exception;

class Message
{
public:
  Message();
  explicit Message( const nlohmann::json& content, const MessageType& messageType =
  MessageType::Unknown );
  explicit Message( const std::string& content, const MessageType& messageType = MessageType::Unknown
  );
  explicit 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:
  explicit Message( const MessageType& messageType );
  void           setContent( const nlohmann::json& content );
  void           setContent( const std::string& content );
  void           setContent( const char* content );
  nlohmann::json m_JSON;
};

class MessageException : public Message
{
public:
  explicit 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();
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_MESSAGETYPE
#define YAODAQ_MESSAGETYPE

#include "yaodaq/Interrupt.hpp"

#include <cstdint>
#include <iosfwd>

namespace yaodaq
{

enum class MessageType : std::int_least16_t
{
  // IXWebSocket MessageType (Message is not set here)
  Open = -6,
  Close,
  Error,
  Ping,
  Pong,
  Fragment,
  // Unknown should not be used !
  Unknown = 0,
  Exception,
};

inline std::ostream& operator«( std::ostream& os, const MessageType& messageTypes ) { return os «
 static_cast<std::int_least8_t>( messageTypes ); }

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_SEVERITY
#define YAODAQ_SEVERITY

#include <cstdint>

namespace yaodaq
{

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

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_SIGNAL
#define YAODAQ_SIGNAL

#include "yaodaq/Severity.hpp"

#include <cstdint>

namespace yaodaq
{

enum class Signal
{
  NO   = 0,  // No Signal.
  // Critical
  ABRT = static_cast<int>( Severity::Critical ) + 1,  // (Signal Abort) Abnormal termination, such as
 is initiated by the abort function.
  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).
  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.
  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.
  // Warning
  INT  = static_cast<int>( Severity::Warning ) + 1,  // (Signal Interrupt) Interactive attention
 signal. Generally generated by the application user.
  TERM = static_cast<int>( Severity::Warning ) + 2,  // (Signal Terminate) Termination request sent to
 program.
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_STATUSCODE
#define YAODAQ_STATUSCODE

#include <cstdint>

namespace yaodaq
{

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

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_VERSION
#define YAODAQ_VERSION

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

namespace yaodaq
{

class Version : public semver::version
{
public:
  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 ) {}
  explicit constexpr Version( const std::string_view& str ) : semver::version( str ) {}
  constexpr Version() = default;
  std::uint8_t getMajor();
  std::uint8_t getMinor();
  std::uint8_t getPatch();
  std::string  getPreRelease();
  std::uint8_t getPreReleaseNumber();
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_WEBSOCKETCLIENT
#define YAODAQ_WEBSOCKETCLIENT

#include "yaodaq/Identifier.hpp"
#include "yaodaq/LoggerHandler.hpp"
#include "yaodaq/Looper.hpp"

#include <ixwebsocket/IXWebSocket.h>
#include <memory>
#include <spdlog/spdlog.h>
#include <string>

namespace yaodaq
{

class Message;
class Open;
class Close;
class Error;
class Ping;
class Pong;
class Fragment;
class MessageException;

class WebsocketClient : public ix::WebSocket
{
public:
  explicit WebsocketClient( const std::string& name, const std::string& type = "YAODAQWebsocketClient"
  );
  virtual ~WebsocketClient();
  void                                start();
  void                                stop();
  void                                loop();
  std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
  static void throwGeneralIfSameName( const bool& );

  // IXWebsocket
  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& );

private:
  void          onRaisingSignal();
  Identifier    m_Identifier;
  LoggerHandler m_Logger;
  Looper        m_Looper;
  static bool m_ThrowGeneralIfSameName;
};

}  // namespace yaodaq

#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.

```
#ifndef YAODAQ_WEBSOCKETSERVER
#define YAODAQ_WEBSOCKETSERVER

#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>

namespace yaodaq
{

class Message;
class Open;
class Close;
class Error;
class Ping;
class Pong;
class Fragment;
class MessageException;

class WebsocketServer : public ix::WebSocketServer
{
public:
  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,
                            const std::size_t& maxConnections =
 ix::SocketServer::kDefaultMaxConnections, const int& handshakeTimeoutSecs =
 ix::WebSocketServer::kDefaultHandShakeTimeoutSecs, const int& addressFamily =
 ix::SocketServer::kDefaultAddressFamily,
                            const std::string& type = "YAODAQWebsocketServer" );
  virtual ~WebsocketServer();
  void loop();
  void start();
  void stop( bool useless = true );
  void listen();

  // IXWebsocket
  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() { return m_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 );

private:
  void                                  addClient( const Identifier&, ix::WebSocket& );
  void                                  removeClient( ix::WebSocket& );
  void                                  onRaisingSignal();
  bool                                  m_isListening{ false };
  Identifier                            m_Identifier;
  LoggerHandler                         m_Logger;
  Looper                                m_Looper;
  bool                                  m_isStopped{ false };
  bool                                  m_isStarted{ false };
  std::map<Identifier, ix::WebSocket&> m_Clients;
  std::mutex                            m_Mutex;
};

}  // namespace yaodaq

#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.

```

#include "yaodaq/ConnectionState.hpp"

#include "yaodaq/Identifier.hpp"

namespace yaodaq
{

std::list<std::pair<std::string, std::string» ConnectionState::m_Ids{};

ConnectionState::ConnectionState() : ix::ConnectionState() {}

ConnectionState::~ConnectionState()
{
  std::lock_guard<std::mutex> guard( m_Mutex );
  m_Ids.remove( m_Pair );
}

void ConnectionState::computeId( const std::string& host, const Identifier& id )
{
  std::lock_guard<std::mutex> guard( m_Mutex );
  m_Pair = std::pair<std::string, std::string>( host, id.getName() );

  if( id.empty() ) { _id = std::to_string( _globalId++ ); }
  else
  {
    std::list<std::pair<std::string, std::string»::iterator found = std::find( m_Ids.begin(),
 m_Ids.end(), m_Pair );
    if( found == m_Ids.end() )
    {
      _id = id.get();
      m_Ids.push_back( m_Pair );
    }
    else
    {
      setTerminated();
    }
  }
}

}  // 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.

```

#include "yaodaq/Exception.hpp"

namespace yaodaq
{

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" };

fmt::text_style Exception::m_Style = { fg( fmt::color::crimson ) | fmt::emphasis::bold };

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(); }

const char* Exception::what() const noexcept { return m_Message.c_str(); }

const char* Exception::description() const noexcept { return m_Description.c_str(); }

std::int_least32_t Exception::code() const noexcept { return m_Code; }

void Exception::constructMessage()
{
  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() ) );
}

}  // 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.

```

#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>

namespace yaodaq
{

bool Identifier::empty() const
{
  if( get() == Identifier().get() ) return true;
  else
    return false;
}

Identifier::Identifier( const std::string& type, const std::string& name ) : m_Type( type ), m_Name(
 name ) {}

void Identifier::generateKey( const Domain& domain, const Class& c_lass, const Family& family ) {
 m_Key = Key( domain, c_lass, family ); }

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

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

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

std::string Identifier::getType() const { return m_Type; }

std::string Identifier::getName() const { return m_Name; }

Key Identifier::getKey() const { return m_Key; }

std::string Identifier::get() const { return fmt::format( "{0}/{1}/{2}/{3}/{4}", getDomain(),
 getClass(), getFamily(), getType(), getName() ); }

Identifier Identifier::parse( const std::string& id )
{
  std::vector<std::string> result;
  std::string              tmp        = id;
  std::string              separator  = "/";
  std::size_t              second_pos = tmp.find( separator );
  while( second_pos != std::string::npos )
  {
    if( 0 != second_pos )
    {
      std::string word = tmp.substr( 0, second_pos - 0 );
      result.push_back( word );
    }
    else
      result.push_back( "" );
    tmp        = tmp.substr( second_pos + separator.length() );
    second_pos = tmp.find( separator );
    if( second_pos == std::string::npos ) result.push_back( tmp );
  }
  if( result.size() == 5 )
  {
    Identifier identifier( result[3], result[4] );
    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()
 );
    return identifier;
  }
  else
  {
    throw Exception( StatusCode::WRONG_NUMBER_PARAMETERS, "Number of parameters in key should be 5
 (Domain/Class/Family/Type/Name) !" );
  }
}

bool Identifier::operator<( const Identifier& identifier ) const { return this->get() <
 identifier.get(); }

}  // 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.

```

#include "yaodaq/Interrupt.hpp"

#include "yaodaq/Signal.hpp"

#include <atomic>
#include <csignal>
#include <mutex>
#include <thread>

namespace yaodaq
{

volatile std::atomic<Signal> Interrupt::m_Signal = Signal::NO;

Interrupt::Interrupt() { init(); }

void Interrupt::restore()
{
  std::signal( SIGTERM, SIG_DFL );
  std::signal( SIGSEGV, SIG_DFL );
  std::signal( SIGINT, SIG_DFL );
  std::signal( SIGILL, SIG_DFL );
  std::signal( SIGABRT, SIG_DFL );
  std::signal( SIGFPE, SIG_DFL );
}

void Interrupt::init()
{
  setSignal( Signal::TERM );
  setSignal( Signal::TERM );
  setSignal( Signal::SEGV );
  setSignal( Signal::INT );
  setSignal( Signal::ILL );
  setSignal( Signal::ABRT );
  setSignal( Signal::FPE );
}

Interrupt::~Interrupt() { restore(); }

Signal Interrupt::getSignal()
{
  if( m_Signal.load() != Signal::NO )
  {
    std::lock_guard<std::mutex> guard( m_mutex );
    init();
  }
  return m_Signal.load();
}

void Interrupt::setSignal( const Signal& signal )
{
  switch( signal )
  {
    case Signal::ABRT: std::signal( SIGABRT, []( int ) -> void { m_Signal.store( Signal::ABRT ); } );
 break;
    case Signal::FPE: std::signal( SIGFPE, []( int ) -> void { m_Signal.store( Signal::FPE ); } );
 break;
    case Signal::ILL: std::signal( SIGILL, []( int ) -> void { m_Signal.store( Signal::ILL ); } );
 break;
    case Signal::SEGV: std::signal( SIGSEGV, []( int ) -> void { m_Signal.store( Signal::SEGV ); } );
 break;
    case Signal::INT: std::signal( SIGINT, []( int ) -> void { m_Signal.store( Signal::INT ); } );
 break;
    case Signal::TERM: std::signal( SIGTERM, []( int ) -> void { m_Signal.store( Signal::TERM ); } );
 break;
    default: break;
  }
}
}  // 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.

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

namespace yaodaq
{

IXMessage::IXMessage( const MessageType& messageType ) : Message( messageType ) {}

IXMessage::IXMessage( const ix::WebSocketMessagePtr& msg ) : Message()
{
  // FIXME
  nlohmann::json json = nlohmann::json::parse( msg->str, nullptr, false );
  if( json.is_discarded() ) { m_JSON["content"] = static_cast<std::string>( msg->str ); }
  else
    m_JSON = json;
  std::cout « m_JSON.dump() « std::endl;
}

void IXMessage::setConnectionStateInfos( std::shared_ptr<ConnectionState>& connectionState )
{
  nlohmann::json j = getContent();
  j["id"]          = connectionState->getId();
  j["remote_ip"]   = connectionState->getRemoteIp();
  j["remote_port"] = connectionState->getRemotePort();
  setContent( j );
}

std::string IXMessage::getId() const { return get()["content"]["id"].get<std::string>(); }

std::string IXMessage::getRemoteIp() const { return get()["content"]["remote_ip"].get<std::string>();
 }

int IXMessage::getRemotePort() const { return get()["content"]["remote_port"].get<int>(); }

// Open
Open::Open( const ix::WebSocketOpenInfo& openInfo ) : IXMessage( MessageType::Open )
{
  nlohmann::json j = getContent();
  j["uri"]         = openInfo.uri;
  j["headers"]     = openInfo.headers;
  j["protocol"]    = openInfo.protocol;
  setContent( j );
}

Open::Open( const ix::WebSocketOpenInfo& openInfo, std::shared_ptr<ConnectionState>& connectionState )
 : Open( openInfo ) { setConnectionStateInfos( connectionState ); }

std::string Open::getURI() const { return get()["content"]["uri"].get<std::string>(); }

std::map<std::string, std::string> Open::getHeaders() const
{
  std::map<std::string, std::string> ret = get()["content"]["headers"].get<std::map<std::string,
 std::string»();
  return ret;
}

std::string Open::getProtocol() const { return get()["content"]["protocol"].get<std::string>(); }

// Close
Close::Close( const ix::WebSocketCloseInfo& closeInfo ) : IXMessage( MessageType::Close )
{
  nlohmann::json j;
  j["code"]   = closeInfo.code;
  j["reason"] = closeInfo.reason;
  j["remote"] = closeInfo.remote;
  setContent( j );
}

Close::Close( const ix::WebSocketCloseInfo& closeInfo, std::shared_ptr<ConnectionState>&
 connectionState ) : Close( closeInfo ) { setConnectionStateInfos( connectionState ); }

std::uint16_t Close::getCode() const { return get()["content"]["code"].get<std::uint16_t>(); }
std::string   Close::getReason() const { return get()["content"]["reason"].get<std::string>(); }
bool          Close::getRemote() const { return get()["content"]["remote"].get<bool>(); }

// Error
Error::Error( const ix::WebSocketErrorInfo& errorInfo ) : IXMessage( MessageType::Error )
{
  nlohmann::json j;
  j["retries"]             = errorInfo.retries;
  j["wait_time"]           = errorInfo.wait_time;
  j["http_status"]         = errorInfo.http_status;
  j["reason"]              = errorInfo.reason;
  j["decompression_error"] = errorInfo.decompressionError;
  setContent( j );
}

Error::Error( const ix::WebSocketErrorInfo& errorInfo, std::shared_ptr<ConnectionState>&
 connectionState ) : Error( errorInfo ) { setConnectionStateInfos( connectionState ); }

std::uint16_t Error::getRetries() const { return get()["content"]["retries"].get<std::uint16_t>(); }

double Error::getWaitTime() const { return get()["content"]["wait_time"].get<double>(); }

int Error::getHttpStatus() const { return get()["content"]["http_status"].get<int>(); }

std::string Error::getReason() const { return get()["content"]["reason"].get<std::string>(); }

bool Error::getDecompressionError() const { return
 get()["content"]["decompression_error"].get<bool>(); }

// Ping
Ping::Ping( const ix::WebSocketMessagePtr& ping ) : IXMessage( MessageType::Ping ) {}

Ping::Ping( const ix::WebSocketMessagePtr& ping, std::shared_ptr<ConnectionState>& connectionState ) :
 Ping( ping ) { setConnectionStateInfos( connectionState ); }

// Pong
Pong::Pong( const ix::WebSocketMessagePtr& pong ) : IXMessage( MessageType::Pong ) {}

Pong::Pong( const ix::WebSocketMessagePtr& pong, std::shared_ptr<ConnectionState>& connectionState ) :
 Pong( pong ) { setConnectionStateInfos( connectionState ); }

// Fragment
Fragment::Fragment( const ix::WebSocketMessagePtr& fragment ) : IXMessage( MessageType::Fragment ) {}

Fragment::Fragment( const ix::WebSocketMessagePtr& fragment, std::shared_ptr<ConnectionState>&
 connectionState ) : Fragment( fragment ) { setConnectionStateInfos( connectionState ); }

}  // 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.

```

#include "yaodaq/Key.hpp"

#include <cstdint>

namespace yaodaq
{
```

```
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 ); }

std::int_least8_t Key::getDomain() const { return ( m_Key » 24 ) & 0xFF; }

std::int_least8_t Key::getClass() const { return ( m_Key » 16 ) & 0xFF; }

std::int_least16_t Key::getFamily() const { return (m_Key)&0xFFFF; }

std::int_least32_t Key::getKey() const { return m_Key; }

}  // 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.

```

#include "yaodaq/LoggerHandler.hpp"

#include "spdlog/spdlog.h"

namespace yaodaq
{

LoggerHandler::LoggerHandler() { init(); }

void LoggerHandler::setName( const std::string& name )
{
  m_Name = name;
  init();
}

LoggerHandler::~LoggerHandler() {}

void LoggerHandler::setVerbosity( const Verbosity& verbosity )
{
  m_Verbosity = verbosity;
  init();
}

void LoggerHandler::init()
{
  m_Logger = std::make_shared<spdlog::logger>( m_Name, std::begin( m_Sinks ), std::end( m_Sinks ) );
  switch( m_Verbosity )
  {
    case Verbosity::Off: m_Logger->set_level( spdlog::level::off ); break;
    case Verbosity::Trace: m_Logger->set_level( spdlog::level::trace ); break;
    case Verbosity::Debug: m_Logger->set_level( spdlog::level::debug ); break;
    case Verbosity::Info: m_Logger->set_level( spdlog::level::info ); break;
    case Verbosity::Warn: m_Logger->set_level( spdlog::level::warn ); break;
    case Verbosity::Error: m_Logger->set_level( spdlog::level::err ); break;
    case Verbosity::Critical: m_Logger->set_level( spdlog::level::critical ); break;
  }
}

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

void LoggerHandler::addSink( const spdlog::sink_ptr& sink )
{
  m_Sinks.push_back( sink );
  init();
}

void LoggerHandler::clearSinks()
{
  m_Sinks.clear();
  init();
}

}  // 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.

```

#include "yaodaq/Looper.hpp"

#include <chrono>
#include <thread>

namespace yaodaq
{

int Looper::m_instance{ 0 };

Interrupt Looper::m_Interrupt{ Interrupt{} };

int Looper::getInstance() { return m_instance; }

void Looper::supressInstance()
{
  if( m_hasBeenSupressed == false )
  {
    m_hasBeenSupressed = true;
    m_instance--;
  }
}

Looper::Looper()
{
  if( m_hasBeenAdded == false )
  {
    m_hasBeenAdded = true;
    ++m_instance;
  }
}

Signal Looper::loop()
{
  static Signal signal{ yaodaq::Signal::NO };
  if( m_instance == 0 )
  {
    do {
      signal = m_Interrupt.getSignal();
      std::this_thread::sleep_for( std::chrono::microseconds( 1 ) );
    } while( signal == yaodaq::Signal::NO );
  }
  return signal;
}

Signal Looper::getSignal() { return m_Interrupt.getSignal(); }

Looper::~Looper()
{
  if( m_hasBeenAdded == true && m_hasBeenSupressed == false )
  {
    m_hasBeenSupressed = true;
    --m_instance;
  }
}

}  // 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.

```

#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>

// Versions numbers
#include <ixwebsocket/IXWebSocketVersion.h>
#include <yaodaq/YaodaqVersion.hpp>

namespace yaodaq
{

Message::Message()
{
  m_JSON["from"];
  m_JSON["to"];
  m_JSON["type"] = magic_enum::enum_name( MessageType::Unknown );
  m_JSON["uuid"] = ix::uuid4();
  m_JSON["content"];
  m_JSON["timestamp"]                        = fmt::format( "{:%F %T %z}", fmt::gmtime(
 std::chrono::system_clock::to_time_t( std::chrono::system_clock::now() ) ) );
  m_JSON["meta"]["compiler"]                 = nlohmann::json::meta()["compiler"];
  m_JSON["meta"]["platform"]                 = nlohmann::json::meta()["platform"];
  m_JSON["meta"]["versions"]["json"]         = nlohmann::json::meta()["version"]["string"];
  m_JSON["meta"]["versions"]["yaodaq"]       = yaodaq_version.to_string();
  m_JSON["meta"]["versions"]["ixwebsocket"] = std::string( IX_WEBSOCKET_VERSION );
}

void Message::setContent( const nlohmann::json& content ) { m_JSON["content"] =
 static_cast<nlohmann::json>( content ); }

void Message::setContent( const std::string& content )
{
  m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
  if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
}

void Message::setContent( const char* content )
{
  m_JSON["content"] = nlohmann::json::parse( content, nullptr, false );
  if( m_JSON["content"].is_discarded() ) { m_JSON["content"] = static_cast<std::string>( content ); }
}

```

```cpp
Message::Message( const nlohmann::json& content, const MessageType& messageType ) : Message( messageType ) { setContent( content ); }

Message::Message( const std::string& content, const MessageType& messageType ) : Message( messageType ) { setContent( content ); }

Message::Message( const char* content, const MessageType& messageType ) : Message( messageType ) { setContent( content ); }

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 ); }

nlohmann::json Message::get() const { return m_JSON; }

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

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

nlohmann::json Message::getContent() const { return m_JSON["content"]; }

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

std::time_t Message::getTime() const
{
  std::tm tm;
  memset( &tm, 0, sizeof( tm ) );
  std::istringstream ss( getTimestamp() );
  ss » std::get_time( &tm, "%Y-%m-%d %H:%M:%S %z" );
  return mktime( &tm );
}

void Message::setFrom( const Identifier& identifier )
{
  m_JSON["from"]["name"]   = identifier.getName();
  m_JSON["from"]["type"]   = identifier.getType();
  m_JSON["from"]["family"] = identifier.getFamily();
  m_JSON["from"]["class"]  = identifier.getClass();
  m_JSON["from"]["domain"] = identifier.getDomain();
}

Identifier Message::getIdentifier() const
{
  if( m_JSON["from"].is_null() ) return {};
  else
  {
    Identifier id( m_JSON["from"]["type"].get<std::string>(), m_JSON["from"]["name"].get<std::string>() );
    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(),
                    magic_enum::enum_cast<Family>( m_JSON["from"]["family"].get<std::string>() ).value() );
    return id;
  }
}

Message::Message( const MessageType& messageType ) : Message() { m_JSON["type"] = magic_enum::enum_name( messageType ); }

// MessageException
MessageException::MessageException( const Exception& exception ) : Message( MessageType::Exception )
{
  nlohmann::json j;
  j["code"]          = exception.code();
  j["description"]   = exception.description();
  j["line"]          = exception.line();
  j["column"]        = exception.column();
  j["file_name"]     = exception.file_name();
  j["function_name"] = exception.function_name();
  setContent( j );
}

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

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

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

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

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

```

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

}  // 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.

```

#include "yaodaq/Version.hpp"

#include <magic_enum.hpp>

namespace yaodaq
{

std::uint8_t Version::getMajor() { return major; }

std::uint8_t Version::getMinor() { return minor; }

std::uint8_t Version::getPatch() { return patch; }

std::string Version::getPreRelease() { return std::string( magic_enum::enum_name( prerelease_type ) );
  }

std::uint8_t Version::getPreReleaseNumber() { return prerelease_number; }

const static Version yaodaq_version;

}  // 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.

```

#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>

namespace yaodaq
{

bool WebsocketClient::m_ThrowGeneralIfSameName{ true };

void WebsocketClient::throwGeneralIfSameName( const bool& activate ) { m_ThrowGeneralIfSameName =
 activate; }

WebsocketClient::WebsocketClient( const std::string& name, const std::string& type ) : m_Identifier(
 type, name )
{
  ix::initNetSystem();

  m_Identifier.generateKey( Domain::Application, Class::Client, Family::WebSocketClient );
  m_Logger.setName( m_Identifier.get() );
  m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );

  ix::WebSocketHttpHeaders header{ { "id", m_Identifier.get() } };
  setExtraHeaders( header );

  setOnMessageCallback(
    [this]( const ix::WebSocketMessagePtr& msg )
    {
      if( msg->type == ix::WebSocketMessageType::Message )
      {
        IXMessage ixmessage( msg );
        onMessage( ixmessage );
      }
      else if( msg->type == ix::WebSocketMessageType::Open )
      {
        Open open( msg->openInfo );
        onOpen( open );
      }
      else if( msg->type == ix::WebSocketMessageType::Close )
      {
        Close close( msg->closeInfo );
        onClose( close );
      }
      else if( msg->type == ix::WebSocketMessageType::Error )
      {
        Error error( msg->errorInfo );
        onError( error );
      }
      else if( msg->type == ix::WebSocketMessageType::Ping )
      {
        Ping ping( msg );
        onPing( ping );
      }
      else if( msg->type == ix::WebSocketMessageType::Pong )
      {
        Pong pong( msg );
        onPong( pong );
      }
      else if( msg->type == ix::WebSocketMessageType::Fragment )
      {
        Fragment fragment( msg );
        onFragment( fragment );
      }
    } );
}

void WebsocketClient::onMessage( Message& message )
{
  switch( message.getTypeValue() )
  {
    case MessageType::Exception:
      MessageException& message_exception = reinterpret_cast<MessageException&>( message );
      // Special case for connection to the server with the same name as an other client !
      if( static_cast<StatusCode>( message_exception.getCode() ) ==
 StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED )
      {
        disableAutomaticReconnection();
        std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
        close( message_exception.getCode(), message_exception.getDescription() );
        std::this_thread::sleep_for( std::chrono::milliseconds( 100 ) );
        if( m_ThrowGeneralIfSameName ) throw Exception( static_cast<StatusCode>( message_exception.getCode() ), message_exception.getDescription() );
      }
      onException( message_exception );
      break;
  }
}

void WebsocketClient::onOpen( Open& open )
{
  std::string headers;
  for( auto it: open.getHeaders() ) { headers += fmt::format( "\t{}: {}\n", it.first, it.second ); }
  logger()->debug( fmt::format( fg( fmt::color::green ), "Connection opened:\nURI: {}\nProtocol: {}\nHeaders:\n{}", open.getURI(), open.getProtocol(), headers ) );
}

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

void WebsocketClient::onError( Error& error )
{
  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() ) );
}

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

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

void WebsocketClient::onFragment( Fragment& fragment ) {}

void WebsocketClient::onException( MessageException& message )
{
  Exception exception( StatusCode::CLIENT_WITH_SAME_NAME_ALREADY_CONNECTED, message.getDescription() );
  logger()->critical( "{Exception }", exception.what() );
}

WebsocketClient::~WebsocketClient()
{
  stop();
  ix::uninitNetSystem();
}

void WebsocketClient::start()
{
  if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
  {
    logger()->trace( "Client started. Connected to {}", getUrl() );
    ix::WebSocket::start();
  }
}

void WebsocketClient::stop()
{
  if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
  {
    logger()->trace( "Client stopped" );
    ix::WebSocket::stop();
    while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for( std::chrono::microseconds( 1 ) ); }
  }
}

void WebsocketClient::loop()
{
  WebsocketClient::start();
  m_Looper.supressInstance();
  onRaisingSignal();
}

void WebsocketClient::onRaisingSignal()
{
  Signal signal = m_Looper.loop();
  if( m_Looper.getInstance() == 0 )
  {
    int value = magic_enum::enum_integer( signal );
    if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical( "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
    else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
    {
      logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
    {
      fmt::print( "\n" );
      logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
    {
      fmt::print( "\n" );
      logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    else
    {
      fmt::print( "\n" );
      logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
       std::exit( magic_enum::enum_integer( signal ) );
  }
}

}  // 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.

```

#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>

namespace yaodaq
{
```

```

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 ) :
  ix::WebSocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ), m_Identifier( type, name )
{
  ix::initNetSystem();

  m_Identifier.generateKey( Domain::Application, Class::Server, Family::WebSocketServer );
  m_Logger.setName( m_Identifier.get() );
  m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );

  setConnectionStateFactory( []() { return std::make_shared<ConnectionState>(); } );

  setOnClientMessageCallback(
    [this]( std::shared_ptr<ix::ConnectionState> connectionState, ix::WebSocket& webSocket, const ix::WebSocketMessagePtr& msg )
    {
      // The ConnectionState object contains information about the connection
      std::shared_ptr<ConnectionState> connection = std::static_pointer_cast<ConnectionState>( connectionState );
      if( msg->type == ix::WebSocketMessageType::Message )
      {
        IXMessage ixmessage( msg );
        // sendToLoggers(ixmessage,webSocket);
        onMessage( ixmessage );
      }
      else if( msg->type == ix::WebSocketMessageType::Open )
      {
        // Check if a client with the same name is already connected;
        connection->computeId( getHost() + ":" + std::to_string( getPort() ), Identifier::parse( msg->openInfo.headers["id"] ) );
        if( connection->isTerminated() )
        {
          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() ) );
          MessageException message_exception(
            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() ) ) );

          // Send to the client on exception;
          webSocket.send( message_exception.dump() );
          // Send to loggers except the client as it has been send before;
          // sendToLoggers(message_exception,webSocket);

          webSocket.stop( magic_enum::enum_integer( 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() ) );
          std::this_thread::sleep_for( std::chrono::milliseconds( 50 ) );
          return;
        }
        addClient( Identifier::parse( msg->openInfo.headers["id"] ), webSocket );
        Open open( msg->openInfo, connection );
        // sendToLoggers( open, webSocket );
        onOpen( open );
      }
      else if( msg->type == ix::WebSocketMessageType::Close )
      {
        Close close( msg->closeInfo, connection );
        // sendToLoggers( close, webSocket );
        onClose( close );
        removeClient( webSocket );
      }
      else if( msg->type == ix::WebSocketMessageType::Error )
      {
        Error error( msg->errorInfo, connection );
        // sendToLoggers( error, webSocket );
        onError( error );
      }
      else if( msg->type == ix::WebSocketMessageType::Ping )
      {
        Ping ping( msg, connection );
        // sendToLoggers( ping, webSocket );
        onPing( ping );
      }
      else if( msg->type == ix::WebSocketMessageType::Pong )
      {
        Pong pong( msg, connection );
        // sendToLoggers( pong, webSocket );
        onPong( pong );
      }
      else if( msg->type == ix::WebSocketMessageType::Fragment )
      {
        Fragment fragment( msg, connection );
        // sendToLoggers( fragment, webSocket );
        onFragment( fragment );
      }
    } );
}

void WebsocketServer::addClient( const Identifier& identifier, ix::WebSocket& websocket )
{
  std::lock_guard<std::mutex> guard( m_Mutex );
  m_Clients.try_emplace( identifier, websocket );
}

void WebsocketServer::removeClient( ix::WebSocket& websocket )
{
  std::lock_guard<std::mutex> guard( m_Mutex );
  for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end(); ++it )
  {
    if( &it->second == &websocket )
    {
      m_Clients.erase( it->first );
      break;
    }
  }
}

void WebsocketServer::sendToLoggers( Message& message, ix::WebSocket& webSocket )
{
  for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end(); ++it )
  {
    if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket != &it->second ) it->second.send( message.dump() );
  }
}

void WebsocketServer::sendToLoggers( Message& message )
{
  for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end(); ++it )
  {
    if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger ) it->second.send( message.dump() );
  }
}

void WebsocketServer::sendToLoggers( const Message& message, ix::WebSocket& webSocket )
{
  for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end(); ++it )
  {
    if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger && &webSocket != &it->second ) it->second.send( message.dump() );
  }
}

void WebsocketServer::sendToLoggers( const Message& message )
{
  for( std::map<Identifier, ix::WebSocket&>::iterator it = m_Clients.begin(); it != m_Clients.end(); ++it )
  {
    if( magic_enum::enum_cast<Family>( it->first.getFamily() ).value() == Family::Logger ) it->second.send( message.dump() );
  }
}

void WebsocketServer::onMessage( Message& message )
{
  switch( message.getTypeValue() )
  {
    case MessageType::Exception:
    {
      MessageException& message_exception = reinterpret_cast<MessageException&>( message );
      onException( message_exception );
    }
    default:
    {
      onUnknown( message );
    }
  }
}

void WebsocketServer::onOpen( Open& open )
{
  std::string headers;
  for( auto it: open.getHeaders() ) { headers += fmt::format( "\t{}: {}\n", it.first, it.second ); }
  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 ) );
}

void WebsocketServer::onClose( Close& close )
{
  logger()->debug(
    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() ) );
}

void WebsocketServer::onError( Error& error )
{
  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(),
                               error.getRetries(), error.getWaitTime(), error.getHttpStatus(), error.getDecompressionError() ) );
}

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

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

void WebsocketServer::onFragment( Fragment& fragment ) {}

void WebsocketServer::onException( MessageException& message ) {}

void WebsocketServer::onUnknown(Message& unknown)
{
  logger()->error( fmt::format( fg( fmt::color::red ), "Unknwown:\n{}", unknown.dump( 2 ) ) );
}

void WebsocketServer::listen()
{
  if( !m_isListening )
  {
    std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
    if( ret.first )
    {
      m_isListening = ret.first;
      logger()->info( "Server listening on {0}:{1}", getHost(), getPort() );
    }
    else
      throw Exception( StatusCode::LISTEN_ERROR, ret.second );
  }
}

void WebsocketServer::start()
{
  if( !m_isStarted )
  {
    m_isStarted = true;
    logger()->trace( "Server started" );
    ix::WebSocketServer::start();
  }
}

void WebsocketServer::stop( bool useless )
{
  if( !m_isStopped )
  {
    m_isStopped = true;
    useless     = !useless;
    logger()->trace( "Server stopped" );
    ix::WebSocketServer::stop();
  }
}

void WebsocketServer::setVerbosity( const yaodaq::LoggerHandler::Verbosity& verbosity ) { m_Logger.setVerbosity( verbosity ); }

WebsocketServer::~WebsocketServer()
{
  stop();
  ix::uninitNetSystem();
}

void WebsocketServer::loop()
{
  listen();
  start();
  m_Looper.supressInstance();
  onRaisingSignal();
}

void WebsocketServer::onRaisingSignal()
{
  Signal signal = m_Looper.loop();
  if( m_Looper.getInstance() == 0 )
  {
    int value = magic_enum::enum_integer( signal );
    if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
 "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
    else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
    {
      logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
    {
      fmt::print( "\n" );
      logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
    {
      fmt::print( "\n" );
      logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    else
    {
      fmt::print( "\n" );
      logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
    }
    if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
 std::exit( magic_enum::enum_integer( signal ) );
  }
}

}  // namespace yaodaq
```