YAODAQ

Yet An Other DAQ

Generated by Doxygen 1.9.3

Chapter 1

License

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 spdlog Namespace Reference

Typedefs

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

6.1.1 Detailed Description

Copyright

Copyright 2022 flagarde

6.1.2 Typedef Documentation

6.1.2.1 sink_ptr

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

Definition at line 15 of file LoggerHandler.hpp.

6.2 yaodaq Namespace Reference

Data Structures

- class Exception
- class Identifier
- class Interrupt
- class LoggerHandler
- class Looper
- class Version
- class WebsocketClient
- class WebsocketServer

Enumerations

- enum class Class : std::int_least16_t {
  Unknown = -1 , Module = 0 , Browser = 100 , WebsocketServer = Module + 1 ,
  WebsocketClient = Module + 2 }
- 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 }

6.2.1 Detailed Description

Copyright

Copyright 2022 flagarde

6.2.2 Enumeration Type Documentation

6.2.2.1 Class

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

Enumerator

Unknown	
Module	
Browser	
WebsocketServer	
WebsocketClient	

Definition at line 13 of file Class.hpp.

```
00014 {
00015   Unknown = -1,
00016   Module  = 0,
00017   Browser = 100,
00018
00019   WebsocketServer = Module + 1,
00020   WebsocketClient = Module + 2,
00021 };
```

6.2.2.2 Severity

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

Enumerator

Info	
Warning	
Error	
Critical	

Definition at line 13 of file Severity.hpp.

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

6.2.2.3 Signal

```
enum class yaodaq::Signal  [strong]
```

Enumerator

NO	
ABRT	
FPE	
ILL	
SEGV	
INT	
TERM	

Definition at line 15 of file Signal.hpp.

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

6.2.2.4 StatusCode

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

Enumerator

SUCCESS	
LISTEN_ERROR	

Definition at line 13 of file StatusCode.hpp.

```
00014 {
00015   SUCCESS = 0,
00016   LISTEN_ERROR,
00017 };
```

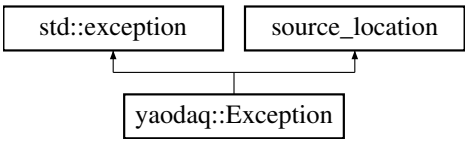
Chapter 7

Data Structure Documentation

7.1 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={})

7.1.1 Detailed Description

Definition at line 19 of file Exception.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Exception() [1/2]

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

7.1.2.2 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(); }
```

7.1.2.3 ~Exception()

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

7.1.3 Member Function Documentation

7.1.3.1 code()

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

Definition at line 20 of file Exception.cpp.

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

7.1.3.2 description()

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

Definition at line 18 of file Exception.cpp.

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

7.1.3.3 setFormat()

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

Definition at line 24 of file Exception.hpp.
```
00024 { m_Format = format; }
```

7.1.3.4 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; }
```

7.1.3.5 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

7.2 yaodaq::Identifier Class Reference

`#include <yaodaq/Identifier.hpp>`

Public Member Functions

- Identifier ()=default
- Identifier (const Class &aClass, const std::string &type, const std::string &name)
- std::string getClass () const
- std::string getType () const
- std::string getName () const
- Class getClassId () const
- std::string get () const

7.2.1 Detailed Description

Definition at line 15 of file Identifier.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Identifier() [1/2]

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

7.2.2.2 Identifier() [2/2]

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

Definition at line 16 of file Identifier.cpp.

```
00016 : m_Class( aClass ), m_Type( type ), m_Name( name ) {}
```

7.2.3 Member Function Documentation

7.2.3.1 get()

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

Definition at line 26 of file Identifier.cpp.

```
00026 { return fmt::format( "{0}/{1}/{2}", getClass(), getType(), getName() ); }
```

7.2.3.2 getClass()

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

Definition at line 18 of file Identifier.cpp.

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

7.2.3.3 getClassId()

```
Class yaodaq::Identifier::getClassId ( ) const
```

Definition at line 24 of file Identifier.cpp.

```
00024 { return m_Class; }
```

7.2.3.4 getName()

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

Definition at line 22 of file Identifier.cpp.
```
00022 { return m_Name; }
```

7.2.3.5 getType()

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

Definition at line 20 of file Identifier.cpp.
```
00020 { return m_Type; }
```

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

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

7.3 yaodaq::Interrupt Class Reference

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

Public Member Functions

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

7.3.1 Detailed Description

Definition at line 19 of file Interrupt.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Interrupt()

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

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

7.3.2.2 ~Interrupt()

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

Definition at line 42 of file Interrupt.cpp.

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

7.3.3 Member Function Documentation

7.3.3.1 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 }
```

7.3.3.2 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 }
```

7.3.3.3 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

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

7.4.1 Detailed Description

Definition at line 21 of file LoggerHandler.hpp.

7.4.2 Member Enumeration Documentation

7.4.2.1 Verbosity

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

Enumerator

Off	
Trace	
Debug	
Info	
Warn	
Error	
Critical	

Definition at line 24 of file LoggerHandler.hpp.

```
00025   {
00026     Off,
00027     Trace,
00028     Debug,
```

```
00029     Info,
00030     Warn,
00031     Error,
00032     Critical
00033   };
```

7.4.3 Constructor & Destructor Documentation

7.4.3.1 LoggerHandler() [1/2]

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

Definition at line 12 of file LoggerHandler.cpp.

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

7.4.3.2 LoggerHandler() [2/2]

```
yaodaq::LoggerHandler::LoggerHandler (
              const std::string & name )  [explicit]
```

Definition at line 14 of file LoggerHandler.cpp.

```
00014 : m_Name( name ) { init(); }
```

7.4.3.3 ~LoggerHandler()

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

Definition at line 16 of file LoggerHandler.cpp.

```
00016 {}
```

7.4.4 Member Function Documentation

7.4.4.1 addSink()

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

Definition at line 41 of file LoggerHandler.cpp.

```
00042 {
00043   m_Sinks.push_back( sink );
00044   init();
00045 }
```

7.4.4.2 clearSinks()

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

Definition at line 47 of file LoggerHandler.cpp.

```
00048 {
00049   m_Sinks.clear();
00050   init();
00051 }
```

7.4.4.3 logger()

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

Definition at line 39 of file LoggerHandler.cpp.

```
00039 { return std::shared_ptr<spdlog::logger>( m_Logger ); }
```

7.4.4.4 setVerbosity()

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

Definition at line 18 of file LoggerHandler.cpp.

```
00019 {
00020   m_Verbosity = verbosity;
00021   init();
00022 }
```

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

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

7.5 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 ()

7.5.1 Detailed Description

Definition at line 15 of file Looper.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Looper()

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

Definition at line 28 of file Looper.cpp.

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

7.5.2.2 ~Looper()

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

Definition at line 52 of file Looper.cpp.

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

7.5.3 Member Function Documentation

7.5.3.1 getInstance()

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

Definition at line 17 of file Looper.cpp.

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

7.5.3.2 getSignal()

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

Definition at line 50 of file Looper.cpp.

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

7.5.3.3 loop()

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

Definition at line 37 of file Looper.cpp.

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

7.5.3.4 supressInstance()

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

Definition at line 19 of file Looper.cpp.

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

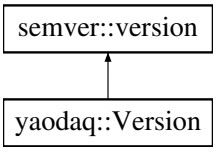
The documentation for this class was generated from the following files:

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

7.6 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 ()

7.6.1 Detailed Description

Definition at line 15 of file Version.hpp.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 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 ) {}
```

7.6.2.2 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 ) {}
```

7.6.2.3 Version() [3/3]

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

7.6.3 Member Function Documentation

7.6.3.1 getMajor()

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

Definition at line 12 of file Version.cpp.

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

7.6.3.2 getMinor()

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

Definition at line 14 of file Version.cpp.

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

7.6.3.3 getPatch()

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

Definition at line 16 of file Version.cpp.

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

7.6.3.4 getPreRelease()

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

Definition at line 18 of file Version.cpp.

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

7.6.3.5 getPreReleaseNumber()

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

Definition at line 20 of file Version.cpp.

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

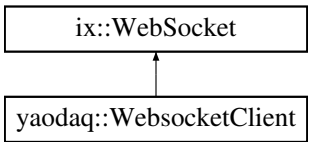
The documentation for this class was generated from the following files:

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

7.7 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="DefaultWebsocketClient")
- virtual ~WebsocketClient ()
- void start ()
- void stop ()
- void loop ()
- std::shared_ptr< spdlog::logger > logger ()

7.7.1 Detailed Description

Definition at line 22 of file WebsocketClient.hpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 WebsocketClient()

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

Definition at line 16 of file WebsocketClient.cpp.

```
00016                                                                   : m_Identifier(
      Class::WebsocketClient, type, name ), m_Logger( m_Identifier.get() )
00017 {
00018   ix::initNetSystem();
00019   ix::WebSocketHttpHeaders header{ { "Id", m_Identifier.get() } };
00020   setExtraHeaders( header );
00021   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00022   setOnMessageCallback(
00023     [this]( const ix::WebSocketMessagePtr& msg )
00024     {
00025       if( msg->type == ix::WebSocketMessageType::Message ) { logger()->error( "{}", msg->str ); }
00026     } );
00027 }
```

7.7.2.2 ~WebsocketClient()

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

Definition at line 29 of file WebsocketClient.cpp.

```
00030 {
00031   stop();
00032   ix::uninitNetSystem();
00033 }
```

7.7.3 Member Function Documentation

7.7.3.1 logger()

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

Definition at line 30 of file WebsocketClient.hpp.

```
00030 { return m_Logger.logger(); }
```

7.7.3.2 loop()

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

Definition at line 54 of file WebsocketClient.cpp.

```
00055 {
00056   WebsocketClient::start();
00057   m_Looper.supressInstance();
00058   onRaisingSignal();
00059 }
```

7.7.3.3 start()

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

Definition at line 35 of file WebsocketClient.cpp.

```
00036 {
00037   if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00038   {
00039     logger()->trace( "Client started. Connected to {}", getUrl() );
00040     ix::WebSocket::start();
00041   }
00042 }
```

7.7.3.4 stop()

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

Definition at line 44 of file WebsocketClient.cpp.

```
00045 {
00046   if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00047   {
00048     logger()->trace( "Client stopped" );
00049     ix::WebSocket::stop();
00050     while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
       std::chrono::microseconds( 1 ) ); }
00051   }
00052 }
```

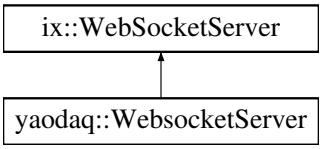
The documentation for this class was generated from the following files:

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

7.8 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="DefaultWebsocketServer")
- virtual ~WebsocketServer ()
- void loop ()
- void start ()
- void stop (bool useless=true)
- void listen ()
- void setVerbosity (const yaodaq::LoggerHandler::Verbosity &verbosity)
- std::shared_ptr< spdlog::logger > logger ()

7.8.1 Detailed Description

Definition at line 22 of file WebsocketServer.hpp.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 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 = "DefaultWebsocketServer" )  [explicit]
```

Definition at line 22 of file WebsocketServer.cpp.

```
00022
                                :
00023   ix::WebSocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
       m_Identifier( Class::WebsocketServer, type, name ), m_Logger( m_Identifier.get() )
00024 {
00025   ix::initNetSystem();
00026   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00027   setOnClientMessageCallback(
00028     []( std::shared_ptr<ix::ConnectionState> connectionState, ix::WebSocket& webSocket, const
      ix::WebSocketMessagePtr& msg )
00029     {
00030       // The ConnectionState object contains information about the connection,
00031       // at this point only the client ip address and the port.
00032       std::cout « "Remote ip: " « connectionState->getRemoteIp() « std::endl;
00033
00034       if( msg->type == ix::WebSocketMessageType::Open )
00035       {
00036         std::cout « "New connection" « std::endl;
00037
00038         // A connection state object is available, and has a default id
00039         // You can subclass ConnectionState and pass an alternate factory
00040         // to override it. It is useful if you want to store custom
00041         // attributes per connection (authenticated bool flag, attributes, etc...)
00042         std::cout « "id: " « connectionState->getId() « std::endl;
00043
00044         // The uri the client did connect to.
00045         std::cout « "Uri: " « msg->openInfo.uri « std::endl;
00046
00047         std::cout « "Headers:" « std::endl;
00048         for( auto it: msg->openInfo.headers ) { std::cout « "\t" « it.first « ": " « it.second «
      std::endl; }
00049       }
00050       else if( msg->type == ix::WebSocketMessageType::Message )
00051       {
00052         // For an echo server, we just send back to the client whatever was received by the server
00053         // All connected clients are available in an std::set. See the broadcast cpp example.
00054         // Second parameter tells whether we are sending the message in binary or text mode.
00055         // Here we send it in the same mode as it was received.
00056         std::cout « "Received: " « msg->str « std::endl;
00057
00058         webSocket.send( msg->str, msg->binary );
00059       }
00060     } );
00061 }
```

7.8.2.2 ~WebsocketServer()

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

Definition at line 101 of file WebsocketServer.cpp.

```
00102 {
00103   stop();
00104   ix::uninitNetSystem();
00105 }
```

7.8.3 Member Function Documentation

7.8.3.1 listen()

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

Definition at line 63 of file WebsocketServer.cpp.

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

7.8.3.2 logger()

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

Definition at line 36 of file WebsocketServer.hpp.

```
00036 { return m_Logger.logger(); }
```

7.8.3.3 loop()

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

Definition at line 107 of file WebsocketServer.cpp.

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

7.8.3.4 setVerbosity()

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

Definition at line 99 of file WebsocketServer.cpp.

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

7.8.3.5 start()

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

Definition at line 78 of file WebsocketServer.cpp.

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

7.8.3.6 stop()

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

Definition at line 88 of file WebsocketServer.cpp.

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

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

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

Chapter 8

File Documentation

8.1 License.md File Reference

8.2 yaodaq/Class.hpp File Reference

```
#include <cstdint>
```

Namespaces

- namespace yaodaq

Enumerations

- enum class yaodaq::Class : std::int_least16_t {
  yaodaq::Unknown = -1 , yaodaq::Module = 0 , yaodaq::Browser = 100 , yaodaq::WebsocketServer = Module + 1 ,
  yaodaq::WebsocketClient = Module + 2 }

8.3 Class.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_CLASS_HPP
00002 #define YAODAQ_CLASS_HPP
00003
00008 #include <cstdint>
00009
00010 namespace yaodaq
00011 {
00012
00013 enum class Class : std::int_least16_t
00014 {
00015   Unknown = -1,
00016   Module  = 0,
00017   Browser = 100,
00018
00019   WebsocketServer = Module + 1,
00020   WebsocketClient = Module + 2,
00021 };
00022
00023 }  // namespace yaodaq
00024
00025 #endif  // YAODAQ_CLASS_HPP
```

8.4 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

8.5 Exception.hpp

Go to the documentation of this file.

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

8.6 yaodaq/Identifier.hpp File Reference

```
#include "yaodaq/Class.hpp"
#include <string>
```

Data Structures

- class yaodaq::Identifier

Namespaces

- namespace yaodaq

8.7 Identifier.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_IDENTIFIER_HPP
00002 #define YAODAQ_IDENTIFIER_HPP
00003
00008 #include "yaodaq/Class.hpp"
00009
00010 #include <string>
00011
00012 namespace yaodaq
00013 {
00014
00015 class Identifier
00016 {
00017 public:
00018   Identifier() = default;
00019   Identifier( const Class& aClass, const std::string& type, const std::string& name );
00020   [[nodiscard]] std::string getClass() const;
00021   [[nodiscard]] std::string getType() const;
00022   [[nodiscard]] std::string getName() const;
00023   [[nodiscard]] Class       getClassId() const;
00024   [[nodiscard]] std::string get() const;
00025
00026 private:
00027   Class       m_Class{ Class::Unknown };
00028   std::string m_Type{ "Unknown" };
00029   std::string m_Name{ "Unknown" };
00030 };
00031
00032 }  // namespace yaodaq
00033
00034 #endif  // YAODAQ_IDENTIFIER_HPP
```

8.8 yaodaq/Interrupt.hpp File Reference

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

Data Structures

- class yaodaq::Interrupt

Namespaces

- namespace yaodaq

8.9 Interrupt.hpp

Go to the documentation of this file.

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

8.10 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 >

8.11 LoggerHandler.hpp

Go to the documentation of this file.

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

8.12 yaodaq/Looper.hpp File Reference

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

Data Structures

- class yaodaq::Looper

Namespaces

- namespace yaodaq

8.13 Looper.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_LOOPER
00002 #define YAODAQ_LOOPER
00003
00008 #include "yaodaq/Interrupt.hpp"
00009
00010 namespace yaodaq
00011 {
00012
00013 enum class Signal;
00014
00015 class Looper
00016 {
00017 public:
00018   Looper();
00019   Signal     loop();
00020   Signal     getSignal();
00021   static int getInstance();
00022   void       supressInstance();
00023   ~Looper();
00024
00025 private:
00026   static int       m_instance;
00027   bool             m_hasBeenAdded{ false };
00028   bool             m_hasBeenSupressed{ false };
00029   static Interrupt m_Interrupt;
00030 };
00031
00032 }  // namespace yaodaq
00033
00034 #endif  // YAODAQ_LOOPER
```

8.14 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 }

8.15 Severity.hpp

Go to the documentation of this file.

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

8.16 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 }

8.17 Signal.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_SIGNAL
00002 #define YAODAQ_SIGNAL
00003
00008 #include "yaodaq/Severity.hpp"
00009
00010 #include <cstdint>
00011
00012 namespace yaodaq
00013 {
00014
00015 enum class Signal
00016 {
00017   NO   = 0,  // No Signal.
00018   // Critical
00019   ABRT = static_cast<int>( Severity::Critical ) + 1,  // (Signal Abort) Abnormal termination, such as is initiated by the abort function.
00020   FPE  = static_cast<int>( Severity::Critical ) + 2,  // (Signal Floating-Point Exception) Erroneous arithmetic operation, such as zero divide or an operation resulting in overflow (not necessarily with a floating-point operation).
00021   ILL  = static_cast<int>( Severity::Critical ) + 3,  // (Signal Illegal Instruction) Invalid function image, such as an illegal instruction. This is generally due to a corruption in the code or to an attempt to execute data.
00022   SEGV = static_cast<int>( Severity::Critical ) + 4,  // (Signal Segmentation Violation) Invalid access to storage: When a program tries to read or write outside the memory it has allocated.
00023   // Warning
00024   INT  = static_cast<int>( Severity::Warning ) + 1,  // (Signal Interrupt) Interactive attention signal. Generally generated by the application user.
00025   TERM = static_cast<int>( Severity::Warning ) + 2,  // (Signal Terminate) Termination request sent to program.
00026 };
00027
00028 }  // namespace yaodaq
00029
00030 #endif  // YAODAQ_CLASS_HPP
```

8.18 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 }

8.19 StatusCode.hpp

Go to the documentation of this file.

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

8.20 yaodaq/Version.hpp File Reference

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

Data Structures

- class yaodaq::Version

Namespaces

- namespace yaodaq

8.21 Version.hpp

Go to the documentation of this file.

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

8.22 yaodaq/WebsocketClient.hpp File Reference

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

Data Structures

- class yaodaq::WebsocketClient

Namespaces

- namespace yaodaq

8.23 WebsocketClient.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_WEBSOCKETCLIENT
00002 #define YAODAQ_WEBSOCKETCLIENT
00003
00008 #include "yaodaq/Identifier.hpp"
00009 #include "yaodaq/Interrupt.hpp"
00010 #include "yaodaq/LoggerHandler.hpp"
00011 #include "yaodaq/Looper.hpp"
00012 #include "yaodaq/YaodaqVersion.hpp"
00013
00014 #include <ixwebsocket/IXWebSocket.h>
00015 #include <memory>
00016 #include <spdlog/spdlog.h>
00017 #include <string>
00018
00019 namespace yaodaq
00020 {
00021
00022 class WebsocketClient : public ix::WebSocket
00023 {
00024 public:
00025   explicit WebsocketClient( const std::string& name, const std::string& type =
       "DefaultWebsocketClient" );
00026   virtual ~WebsocketClient();
00027   void                              start();
00028   void                              stop();
00029   void                              loop();
00030   std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
00031
00032 private:
00033   void          onRaisingSignal();
00034   Identifier    m_Identifier;
00035   LoggerHandler m_Logger;
00036   Looper        m_Looper;
00037 };
00038
00039 }  // namespace yaodaq
00040
00041 #endif
```

8.24 yaodaq/WebsocketServer.hpp File Reference

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

Data Structures

- class yaodaq::WebsocketServer

Namespaces

- namespace yaodaq

8.25 WebsocketServer.hpp

Go to the documentation of this file.

```
00001 #ifndef YAODAQ_WEBSOCKETSERVER
00002 #define YAODAQ_WEBSOCKETSERVER
00003
00008 #include "yaodaq/Identifier.hpp"
00009 #include "yaodaq/Interrupt.hpp"
00010 #include "yaodaq/LoggerHandler.hpp"
00011 #include "yaodaq/Looper.hpp"
00012 #include "yaodaq/YaodaqVersion.hpp"
00013
00014 #include <ixwebsocket/IXWebSocketServer.h>
00015 #include <memory>
00016 #include <spdlog/spdlog.h>
00017 #include <string>
00018
00019 namespace yaodaq
00020 {
00021
00022 class WebsocketServer : public ix::WebSocketServer
00023 {
00024 public:
00025   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,
00026                             const std::size_t& maxConnections =
       ix::SocketServer::kDefaultMaxConnections, const int& handshakeTimeoutSecs =
       ix::WebSocketServer::kDefaultHandShakeTimeoutSecs, const int& addressFamily =
       ix::SocketServer::kDefaultAddressFamily,
00027                             const std::string& type = "DefaultWebsocketServer" );
00028   virtual ~WebsocketServer();
00029   void loop();
00030   void start();
00031   void stop( bool useless = true );
00032   void listen();
00033
00034   void setVerbosity( const yaodaq::LoggerHandler::Verbosity& verbosity );
00035
00036   std::shared_ptr<spdlog::logger> logger() { return m_Logger.logger(); }
00037
00038 private:
00039   void          onRaisingSignal();
00040   bool          m_isListening{ false };
00041   Identifier    m_Identifier;
00042   LoggerHandler m_Logger;
00043   Interrupt     m_Interrupt;
00044   Looper        m_Looper;
00045   bool          m_isStopped{ false };
00046   bool          m_isStarted{ false };
00047 };
00048
00049 }  // namespace yaodaq
00050
00051 #endif  // YAODAQ_WEBSOCKETSERVER
```

8.26 yaodaq/Exception.cpp File Reference

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

Namespaces

- namespace yaodaq

8.27 Exception.cpp

Go to the documentation of this file.

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

8.28 yaodaq/Identifier.cpp File Reference

```
#include "yaodaq/Identifier.hpp"
#include "yaodaq/Class.hpp"
#include <fmt/color.h>
#include <magic_enum.hpp>
#include <string>
```

Namespaces

- namespace yaodaq

8.29 Identifier.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/Identifier.hpp"
00006
00007 #include "yaodaq/Class.hpp"
00008
00009 #include <fmt/color.h>
00010 #include <magic_enum.hpp>
00011 #include <string>
00012
00013 namespace yaodaq
00014 {
00015
00016 Identifier::Identifier( const Class& aClass, const std::string& type, const std::string& name ) :
       m_Class( aClass ), m_Type( type ), m_Name( name ) {}
00017
00018 std::string Identifier::getClass() const { return std::string( magic_enum::enum_name( m_Class ) ); }
00019
00020 std::string Identifier::getType() const { return m_Type; }
00021
00022 std::string Identifier::getName() const { return m_Name; }
00023
00024 Class Identifier::getClassId() const { return m_Class; }
00025
00026 std::string Identifier::get() const { return fmt::format( "{0}/{1}/{2}", getClass(), getType(),
       getName() ); }
00027
00028 }  // namespace yaodaq
```

8.30 yaodaq/Interrupt.cpp File Reference

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

Namespaces

- namespace yaodaq

8.31 Interrupt.cpp

Go to the documentation of this file.

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

```
00058     case Signal::ABRT: std::signal( SIGABRT, []( int ) -> void { m_Signal.store( Signal::ABRT ); } );
      break;
00059     case Signal::FPE: std::signal( SIGFPE, []( int ) -> void { m_Signal.store( Signal::FPE ); } );
      break;
00060     case Signal::ILL: std::signal( SIGILL, []( int ) -> void { m_Signal.store( Signal::ILL ); } );
      break;
00061     case Signal::SEGV: std::signal( SIGSEGV, []( int ) -> void { m_Signal.store( Signal::SEGV ); } );
      break;
00062     case Signal::INT: std::signal( SIGINT, []( int ) -> void { m_Signal.store( Signal::INT ); } );
      break;
00063     case Signal::TERM: std::signal( SIGTERM, []( int ) -> void { m_Signal.store( Signal::TERM ); } );
      break;
00064     default: break;
00065   }
00066 }
00067
00068 }  // namespace yaodaq
```

8.32 yaodaq/LoggerHandler.cpp File Reference

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

Namespaces

- namespace yaodaq

8.33 LoggerHandler.cpp

Go to the documentation of this file.

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

```
00042 {
00043   m_Sinks.push_back( sink );
00044   init();
00045 }
00046
00047 void LoggerHandler::clearSinks()
00048 {
00049   m_Sinks.clear();
00050   init();
00051 }
00052
00053 }  // namespace yaodaq
```

8.34 yaodaq/Looper.cpp File Reference

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

Namespaces

- namespace yaodaq

8.35 Looper.cpp

Go to the documentation of this file.

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

```
00047   return signal;
00048 }
00049
00050 Signal Looper::getSignal() { return m_Interrupt.getSignal(); }
00051
00052 Looper::~Looper()
00053 {
00054   if( m_hasBeenAdded == true && m_hasBeenSupressed == false )
00055   {
00056     m_hasBeenSupressed = true;
00057     --m_instance;
00058   }
00059 }
00060
00061 }  // namespace yaodaq
```

8.36 yaodaq/Version.cpp File Reference

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

Namespaces

- namespace yaodaq

8.37 Version.cpp

Go to the documentation of this file.

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

8.38 yaodaq/WebsocketClient.cpp File Reference

```
#include "yaodaq/WebsocketClient.hpp"
#include <chrono>
#include <ixwebsocket/IXNetSystem.h>
#include <magic_enum.hpp>
#include <spdlog/sinks/stdout_color_sinks.h>
#include <thread>
```

Namespaces

- namespace yaodaq

8.39 WebsocketClient.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/WebsocketClient.hpp"
00006
00007 #include <chrono>
00008 #include <ixwebsocket/IXNetSystem.h>
00009 #include <magic_enum.hpp>
00010 #include <spdlog/sinks/stdout_color_sinks.h>
00011 #include <thread>
00012
00013 namespace yaodaq
00014 {
00015
00016 WebsocketClient::WebsocketClient( const std::string& name, const std::string& type ) : m_Identifier(
       Class::WebsocketClient, type, name ), m_Logger( m_Identifier.get() )
00017 {
00018   ix::initNetSystem();
00019   ix::WebSocketHttpHeaders header{ { "Id", m_Identifier.get() } };
00020   setExtraHeaders( header );
00021   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00022   setOnMessageCallback(
00023     [this]( const ix::WebSocketMessagePtr& msg )
00024     {
00025       if( msg->type == ix::WebSocketMessageType::Message ) { logger()->error( "{}", msg->str ); }
00026     } );
00027 }
00028
00029 WebsocketClient::~WebsocketClient()
00030 {
00031   stop();
00032   ix::uninitNetSystem();
00033 }
00034
00035 void WebsocketClient::start()
00036 {
00037   if( getReadyState() == ix::ReadyState::Closed || getReadyState() == ix::ReadyState::Closing )
00038   {
00039     logger()->trace( "Client started. Connected to {}", getUrl() );
00040     ix::WebSocket::start();
00041   }
00042 }
00043
00044 void WebsocketClient::stop()
00045 {
00046   if( getReadyState() == ix::ReadyState::Open || getReadyState() == ix::ReadyState::Connecting )
00047   {
00048     logger()->trace( "Client stopped" );
00049     ix::WebSocket::stop();
00050     while( getReadyState() != ix::ReadyState::Closed ) { std::this_thread::sleep_for(
       std::chrono::microseconds( 1 ) ); }
00051   }
00052 }
00053
00054 void WebsocketClient::loop()
00055 {
00056   WebsocketClient::start();
00057   m_Looper.supressInstance();
00058   onRaisingSignal();
00059 }
00060
00061 void WebsocketClient::onRaisingSignal()
00062 {
00063   Signal signal = m_Looper.loop();
00064   if( m_Looper.getInstance() == 0 )
00065   {
00066     int value = magic_enum::enum_integer( signal );
00067     if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
       "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00068     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
00069     {
00070       logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00071     }
00072     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
00073     {
```

```
00074       fmt::print( "\n" );
00075       logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00076     }
00077     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
00078     {
00079       fmt::print( "\n" );
00080       logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00081     }
00082     else
00083     {
00084       fmt::print( "\n" );
00085       logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00086     }
00087     if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
       std::exit( magic_enum::enum_integer( signal ) );
00088   }
00089 }
00090
00091 }  // namespace yaodaq
```

8.40 yaodaq/WebsocketServer.cpp File Reference

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

8.41 WebsocketServer.cpp

Go to the documentation of this file.

```
00001
00005 #include "yaodaq/WebsocketServer.hpp"
00006
00007 #include "yaodaq/Exception.hpp"
00008 #include "yaodaq/StatusCode.hpp"
00009
00010 #include <iostream>
00011 #include <ixwebsocket/IXNetSystem.h>
00012 #include <magic_enum.hpp>
00013 #include <spdlog/sinks/stdout_color_sinks.h>
00014 #include <spdlog/spdlog.h>
00015 #include <string>
00016 #include <thread>
00017 #include <utility>
00018
00019 namespace yaodaq
00020 {
00021
00022 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 ) :
00023   ix::WebSocketServer( port, host, backlog, maxConnections, handshakeTimeoutSecs, addressFamily ),
       m_Identifier( Class::WebsocketServer, type, name ), m_Logger( m_Identifier.get() )
00024 {
```

```
00025   ix::initNetSystem();
00026   m_Logger.addSink( std::make_shared<spdlog::sinks::stdout_color_sink_mt>() );
00027   setOnClientMessageCallback(
00028     []( std::shared_ptr<ix::ConnectionState> connectionState, ix::WebSocket& webSocket, const
      ix::WebSocketMessagePtr& msg )
00029     {
00030       // The ConnectionState object contains information about the connection,
00031       // at this point only the client ip address and the port.
00032       std::cout « "Remote ip: " « connectionState->getRemoteIp() « std::endl;
00033 
00034       if( msg->type == ix::WebSocketMessageType::Open )
00035       {
00036         std::cout « "New connection" « std::endl;
00037 
00038         // A connection state object is available, and has a default id
00039         // You can subclass ConnectionState and pass an alternate factory
00040         // to override it. It is useful if you want to store custom
00041         // attributes per connection (authenticated bool flag, attributes, etc...)
00042         std::cout « "id: " « connectionState->getId() « std::endl;
00043 
00044         // The uri the client did connect to.
00045         std::cout « "Uri: " « msg->openInfo.uri « std::endl;
00046 
00047         std::cout « "Headers:" « std::endl;
00048         for( auto it: msg->openInfo.headers ) { std::cout « "\t" « it.first « ": " « it.second «
      std::endl; }
00049       }
00050       else if( msg->type == ix::WebSocketMessageType::Message )
00051       {
00052         // For an echo server, we just send back to the client whatever was received by the server
00053         // All connected clients are available in an std::set. See the broadcast cpp example.
00054         // Second parameter tells whether we are sending the message in binary or text mode.
00055         // Here we send it in the same mode as it was received.
00056         std::cout « "Received: " « msg->str « std::endl;
00057 
00058         webSocket.send( msg->str, msg->binary );
00059       }
00060     } );
00061 }
00062 
00063 void WebsocketServer::listen()
00064 {
00065   if( !m_isListening )
00066   {
00067     std::pair<bool, std::string> ret = ix::WebSocketServer::listen();
00068     if( ret.first )
00069     {
00070       m_isListening = ret.first;
00071       logger()->info( "Server listening on host {0} port {1}", getHost(), getPort() );
00072     }
00073     else
00074       throw Exception( StatusCode::LISTEN_ERROR, ret.second );
00075   }
00076 }
00077 
00078 void WebsocketServer::start()
00079 {
00080   if( !m_isStarted )
00081   {
00082     m_isStarted = true;
00083     logger()->trace( "Server started" );
00084     ix::WebSocketServer::start();
00085   }
00086 }
00087 
00088 void WebsocketServer::stop( bool useless )
00089 {
00090   if( !m_isStopped )
00091   {
00092     m_isStopped = true;
00093     useless     = !useless;
00094     logger()->trace( "Server stopped" );
00095     ix::WebSocketServer::stop();
00096   }
00097 }
00098 
00099 void WebsocketServer::setVerbosity( const yaodaq::LoggerHandler::Verbosity& verbosity ) {
      m_Logger.setVerbosity( verbosity ); }
00100 
00101 WebsocketServer::~WebsocketServer()
00102 {
00103   stop();
00104   ix::uninitNetSystem();
00105 }
00106 
00107 void WebsocketServer::loop()
00108 {
```

```
00109   listen();
00110   start();
00111   m_Looper.supressInstance();
00112   onRaisingSignal();
00113 }
00114
00115 void WebsocketServer::onRaisingSignal()
00116 {
00117   Signal signal = m_Looper.loop();
00118   if( m_Looper.getInstance() == 0 )
00119   {
00120     int value = magic_enum::enum_integer( signal );
00121     if( value >= magic_enum::enum_integer( yaodaq::Severity::Critical ) ) { logger()->critical(
       "Signal SIG{} raised !", magic_enum::enum_name( signal ) ); }
00122     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Error ) )
00123     {
00124       logger()->error( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00125     }
00126     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Warning ) )
00127     {
00128       fmt::print( "\n" );
00129       logger()->warn( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00130     }
00131     else if( value >= magic_enum::enum_integer( yaodaq::Severity::Info ) )
00132     {
00133       fmt::print( "\n" );
00134       logger()->info( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00135     }
00136     else
00137     {
00138       fmt::print( "\n" );
00139       logger()->trace( "Signal SIG{} raised !", magic_enum::enum_name( signal ) );
00140     }
00141     if( magic_enum::enum_integer( signal ) >= magic_enum::enum_integer( Severity::Critical ) )
       std::exit( magic_enum::enum_integer( signal ) );
00142   }
00143 }
00144
00145 }  // namespace yaodaq
```