ZeroMQ Component Model

Generated by Doxygen 1.8.6

Thu Jun 30 2016 14:00:35

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 zcm Namespace Reference

Classes

- class Actor

  Actor class.
- class Client

  Client class.
- class Component

  Component class.
- class Operation_Queue

  Operation_Queue class.
- class Base_Operation

  Base Operation class.
- class Timer_Operation

  Timer Operation class.
- class Subscriber_Operation

  Subscriber Operation class.
- class Server_Operation

  Server Operation class.
- class Publisher

  Publisher class.
- class Server

  Server class.
- class Subscriber

  Subscriber class.
- class Timer

  Timer class.

Chapter 6

Class Documentation

6.1 zcm::Actor Class Reference

Actor class.

`#include <actor.hpp>`

Public Member Functions

- void configure (std::string configuration_file)

  Configure the component_instances vector.
- void run ()

  Spawn all component instances.
- std::string get_name ()

  Get actor name.

Private Attributes

- std::string name
- std::vector< Component * > component_instances

6.1.1 Detailed Description

Actor class.

6.1.2 Member Function Documentation

6.1.2.1 void zcm::Actor::configure (std::string *configuration_file*)

Configure the component_instances vector.

Parameters

`in`	*configuration_file*	JSON configuration file to parse

6.1.2.2 std::string zcm::Actor::get_name ( )

Get actor name.

Returns

Name of the actor

6.1.2.3 void zcm::Actor::run ( )

Spawn all component instances.

6.1.3 Member Data Documentation

6.1.3.1 std::vector<**Component**∗> zcm::Actor::component_instances `[private]`

6.1.3.2 std::string zcm::Actor::name `[private]`

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

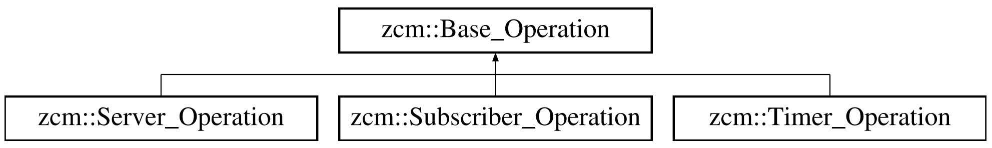
- /home/kelsier/GitHub/zcm/include/actor.hpp
- /home/kelsier/GitHub/zcm/src/actor.cpp

6.2 zcm::Base_Operation Class Reference

Base Operation class.

`#include <operation_types.hpp>`

Inheritance diagram for zcm::Base_Operation:



Public Member Functions

- Base_Operation (std::string name, unsigned int priority)

  Construct a base operation.
- std::string get_name ()

  Return the operation name.
- unsigned int get_priority () const

  Return the operation priority.
- virtual void execute ()

  Virtual execute function overridden by concrete types.

Private Attributes

- std::string name

  Name of the Operation.
- unsigned int priority

  Priority of the Operation.

6.2.1 Detailed Description

Base Operation class.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 zcm::Base_Operation::Base_Operation (std::string *name*, unsigned int *priority*) `[inline]`

Construct a base operation.

Parameters

`in`	*name*	Name of the operation
`in`	*priority*	Priority of the operation

6.2.3 Member Function Documentation

6.2.3.1 virtual void zcm::Base_Operation::execute ( ) `[inline]`, `[virtual]`

Virtual execute function overridden by concrete types.

Reimplemented in zcm::Server_Operation, zcm::Subscriber_Operation, and zcm::Timer_Operation.

6.2.3.2 std::string zcm::Base_Operation::get_name ( )

Return the operation name.

Returns

Name of the operation

6.2.3.3 unsigned int zcm::Base_Operation::get_priority ( ) const

Return the operation priority.

Returns

Priority of the operation

6.2.4 Member Data Documentation

6.2.4.1 std::string zcm::Base_Operation::name `[private]`

Name of the Operation.

6.2.4.2 unsigned int zcm::Base_Operation::priority `[private]`

Priority of the Operation.

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

- /home/kelsier/GitHub/zcm/include/operation_types.hpp
- /home/kelsier/GitHub/zcm/src/operation_types.cpp

6.3 zcm::Client Class Reference

Client class.

```
#include <client.hpp>
```

Public Member Functions

- Client (std::string name, zmq::context_t ∗actor_context, int timeout)

  Construct a client object.
- Client (std::string name, zmq::context_t ∗actor_context, std::vector< std::string > endpoints, int timeout)

  Construct a client object with known endpoints.
- ∼Client ()

  Close the client ZMQ socket and destroy the context.
- void connect (std::vector< std::string > new_endpoints)

  Connect the client to a new set of endpoints.
- std::string get_name ()

  Return the client name.
- void set_timeout (int timeout)

  Set timeout on the client to prevent endless blocking.
- std::string call (std::string message)

  Call the server.

Private Attributes

- std::string name

  Name of the publisher.
- std::vector< std::string > endpoints

  Vector of endpoints to connect to.
- zmq::context_t ∗ context

  ZMQ Context of the client.
- zmq::socket_t ∗ client_socket

  ZMQ Socket of the client.
- int client_socket_timeout

  Timeout of the client socket.

6.3.1 Detailed Description

Client class.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 zcm::Client::Client (std::string *name,* zmq::context_t ∗ *actor_context,* int *timeout* = `500`)

Construct a client object.

Parameters

in	*name*	Client name
in	*ZMQ*	Context of the Actor Process
in	*timeout*	Client socket timeout

6.3.2.2 zcm::Client::Client (std::string *name,* zmq::context_t $*$ *actor_context,* std::vector< std::string > *endpoints,* int *timeout =* `500`)

Construct a client object with known endpoints.

Parameters

in	*name*	Client name
in	*ZMQ*	Context of the Actor Process
in	*endpoints*	A vector of endpoint strings
in	*timeout*	Client socket timeout

6.3.2.3 zcm::Client::∼Client ( )

Close the client ZMQ socket and destroy the context.

6.3.3 Member Function Documentation

6.3.3.1 std::string zcm::Client::call (std::string *message*)

Call the server.

Parameters

in	*message*	The message string. Serialize complex objects to strings with protobuf

6.3.3.2 void zcm::Client::connect (std::vector< std::string > *new_endpoints*)

Connect the client to a new set of endpoints.

Parameters

in	*new_endpoints*	New set of endpoints as a vector

6.3.3.3 std::string zcm::Client::get_name ( )

Return the client name.

Returns

Client name

6.3.3.4 void zcm::Client::set_timeout (int *timeout*)

Set timeout on the client to prevent endless blocking.

Parameters

in	*timeout*	New timeout value

6.3.4 Member Data Documentation

6.3.4.1 zmq::socket_t∗ zcm::Client::client_socket [private]

ZMQ Socket of the client.

6.3.4.2 int zcm::Client::client_socket_timeout [private]

Timeout of the client socket.

6.3.4.3 zmq::context_t∗ zcm::Client::context [private]

ZMQ Context of the client.

6.3.4.4 std::vector<std::string> zcm::Client::endpoints [private]

Vector of endpoints to connect to.

6.3.4.5 std::string zcm::Client::name [private]

Name of the publisher.

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

- /home/kelsier/GitHub/zcm/include/client.hpp
- /home/kelsier/GitHub/zcm/src/client.cpp

6.4 zcm::Component Class Reference

Component class.

`#include <component.hpp>`

Public Member Functions

- Component ()
  Construct a component Prepare the component operation queue.
- ∼Component ()
  Destroy the component.
- Operation_Queue ∗ get_operation_queue ()
  Get pointer to operation_queue.
- Timer ∗ timer (std::string timer_name)
  Get a component timer by name.
- Publisher ∗ publisher (std::string publisher_name)
  Get a component publisher by name.
- Subscriber ∗ subscriber (std::string subscriber_name)
  Get a component subscriber by name.

- Client ∗ client (std::string client_name)

  Get a component client by name.
- Server ∗ server (std::string server_name)

  Get a component server by name.
- void add_timer (Timer ∗new_timer)

  Add a timer to this component.
- void add_publisher (Publisher ∗new_publisher)

  Add a publisher to this component.
- void add_subscriber (Subscriber ∗new_subscriber)

  Add a subscriber to this component.
- void add_client (Client ∗new_client)

  Add a client to this component.
- void add_server (Server ∗new_server)

  Add a server to this component.
- void configure_publishers (std::map< std::string, std::vector< std::string >> publisher_endpoints)

  Configure all component publishers.
- void configure_subscribers (std::map< std::string, std::vector< std::string >> subscriber_endpoints)

  Configure all component subscribers.
- void configure_clients (std::map< std::string, std::vector< std::string >> client_endpoints)

  Configure all component clients.
- void configure_servers (std::map< std::string, std::vector< std::string >> server_endpoints)

  Configure all component servers.
- void register_functionality (std::string operation_name, std::function< void()> operation_function)

  Register component functionality.
- std::thread ∗ spawn ()

  Spawn the component executor thread.

Public Attributes

- std::map< std::string,
  std::function< void()> > functionality

  A map of all component operations.

Protected Attributes

- Operation_Queue ∗ operation_queue

  Pointer to the Component Operation Queue.
- std::thread ∗ executor_thread

  Pointer to the Component Executor Thread.
- std::vector< Timer ∗ > timers

  A vector of component timers.
- std::vector< Publisher ∗ > publishers

  A vector of component publishers.
- std::vector< Subscriber ∗ > subscribers

  A vector of component subscribers.
- std::vector< Client ∗ > clients

  A vector of component clients.
- std::vector< Server ∗ > servers

  A vector of component servers.

6.4.1 Detailed Description

Component class.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 zcm::Component::Component ( )

Construct a component Prepare the component operation queue.

6.4.2.2 zcm::Component::∼Component ( )

Destroy the component.

6.4.3 Member Function Documentation

6.4.3.1 void zcm::Component::add_client (**Client** ∗ *new_client*)

Add a client to this component.

Parameters

`in`	*new_client*	Pointer to a client object

6.4.3.2 void zcm::Component::add_publisher (**Publisher** ∗ *new_publisher*)

Add a publisher to this component.

Parameters

`in`	*new_publisher*	Pointer to a publisher object

6.4.3.3 void zcm::Component::add_server (**Server** ∗ *new_server*)

Add a server to this component.

Parameters

`in`	*new_server*	Pointer to a server object

6.4.3.4 void zcm::Component::add_subscriber (**Subscriber** ∗ *new_subscriber*)

Add a subscriber to this component.

Parameters

`in`	*new_subscriber*	Pointer to a subscriber object

6.4.3.5 void zcm::Component::add_timer (**Timer** ∗ *new_timer*)

Add a timer to this component.

Parameters

`in`	*new_timer*	Pointer to a timer object

6.4.3.6 Client ∗ zcm::Component::client (std::string *client_name*)

Get a component client by name.

Parameters

`in`	*client_name*	Name of the client

6.4.3.7 void zcm::Component::configure_clients (std::map< std::string, std::vector< std::string >> *client_endpoints*)

Configure all component clients.

Parameters

`in`	*client_endpoints*	A map of endpoints for all clients

6.4.3.8 void zcm::Component::configure_publishers (std::map< std::string, std::vector< std::string >> *publisher_endpoints*)

Configure all component publishers.

Parameters

`in`	*publisher_-endpoints*	A map of endpoints for all publishers

6.4.3.9 void zcm::Component::configure_servers (std::map< std::string, std::vector< std::string >> *server_endpoints*)

Configure all component servers.

Parameters

`in`	*server_-endpoints*	A map of endpoints for all servers

6.4.3.10 void zcm::Component::configure_subscribers (std::map< std::string, std::vector< std::string >> *subscriber_endpoints*)

Configure all component subscribers.

Parameters

`in`	*subscriber_-endpoints*	A map of endpoints for all subscribers

6.4.3.11 Operation_Queue ∗ zcm::Component::get_operation_queue ( )

Get pointer to operation_queue.

Returns

Operation Queue pointer

6.4.3.12 Publisher ∗ zcm::Component::publisher (std::string *publisher_name*)

Get a component publisher by name.

Parameters

`in`	*publisher_name*	Name of the publisher

6.4.3.13 void zcm::Component::register_functionality (std::string *operation_name,* std::function< void()> *operation_function*)

Register component functionality.

Parameters

`in`	*operation_name*	Name of the operation
`in`	*operation_-function*	The actual operation function

6.4.3.14 Server ∗ zcm::Component::server (std::string *server_name*)

Get a component server by name.

Parameters

`in`	*server_name*	Name of the server

6.4.3.15 std::thread ∗ zcm::Component::spawn ( )

Spawn the component executor thread.

Returns

Return a pointer to the executor thread

6.4.3.16 Subscriber ∗ zcm::Component::subscriber (std::string *subscriber_name*)

Get a component subscriber by name.

Parameters

`in`	*subscriber_-name*	Name of the subscriber

6.4.3.17 Timer ∗ zcm::Component::timer (std::string *timer_name*)

Get a component timer by name.

Parameters

in	*timer_name*	Name of the timer

6.4.4 Member Data Documentation

6.4.4.1 std::vector<Client∗> zcm::Component::clients [protected]

A vector of component clients.

6.4.4.2 std::thread∗ zcm::Component::executor_thread [protected]

Pointer to the Component Executor Thread.

6.4.4.3 std::map<std::string, std::function<void()> > zcm::Component::functionality

A map of all component operations.

6.4.4.4 Operation_Queue∗ zcm::Component::operation_queue [protected]

Pointer to the Component Operation Queue.

6.4.4.5 std::vector<Publisher∗> zcm::Component::publishers [protected]

A vector of component publishers.

6.4.4.6 std::vector<Server∗> zcm::Component::servers [protected]

A vector of component servers.

6.4.4.7 std::vector<Subscriber∗> zcm::Component::subscribers [protected]

A vector of component subscribers.

6.4.4.8 std::vector<Timer∗> zcm::Component::timers [protected]

A vector of component timers.

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

- /home/kelsier/GitHub/zcm/include/component.hpp
- /home/kelsier/GitHub/zcm/src/component.cpp

6.5 zcm::Operation_Queue Class Reference

Operation_Queue class.

`#include <operation_queue.hpp>`

Classes

- struct PriorityOrdering

Public Member Functions

- void enqueue (Base_Operation $*$new_operation)
- void dequeue ()
- bool empty ()
- Base_Operation $*$ top ()
- void process ()
- std::thread $*$ spawn ()

Private Attributes

- std::priority_queue
  $<$ Base_Operation, std::vector
  $<$ Base_Operation $*$ $>$
  , PriorityOrdering $>$ operation_queue

  The component operation queue - STL priority_queue with fixed-priority scheduling.
- std::mutex queue_mutex

  Mutex that protects the queue during enqueue/dequeue.

6.5.1 Detailed Description

Operation_Queue class.

6.5.2 Member Function Documentation

6.5.2.1 void zcm::Operation_Queue::dequeue ( )

6.5.2.2 bool zcm::Operation_Queue::empty ( )

6.5.2.3 void zcm::Operation_Queue::enqueue (Base_Operation $*$ *new_operation*)

6.5.2.4 void zcm::Operation_Queue::process ( )

6.5.2.5 std::thread $*$ zcm::Operation_Queue::spawn ( )

6.5.2.6 Base_Operation $*$ zcm::Operation_Queue::top ( )

6.5.3 Member Data Documentation

6.5.3.1 std::priority_queue$<$Base_Operation, std::vector$<$Base_Operation$*$$>$, PriorityOrdering$>$ zcm::Operation_Queue::operation_queue `[private]`

The component operation queue - STL priority_queue with fixed-priority scheduling.

6.5.3.2 std::mutex zcm::Operation_Queue::queue_mutex `[private]`

Mutex that protects the queue during enqueue/dequeue.

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

- /home/kelsier/GitHub/zcm/include/operation_queue.hpp
- /home/kelsier/GitHub/zcm/src/operation_queue.cpp

6.6 zcm::Operation_Queue::PriorityOrdering Struct Reference

```
#include <operation_queue.hpp>
```

Public Member Functions

- bool operator() (const Base_Operation ∗lhs, const Base_Operation ∗rhs) const

6.6.1 Member Function Documentation

6.6.1.1 bool zcm::Operation_Queue::PriorityOrdering::operator() (const Base_Operation ∗ *lhs*, const Base_Operation ∗ *rhs*) const [inline]

The documentation for this struct was generated from the following file:

- /home/kelsier/GitHub/zcm/include/operation_queue.hpp

6.7 zcm::Publisher Class Reference

Publisher class.

```
#include <publisher.hpp>
```

Public Member Functions

- Publisher (std::string name, zmq::context_t ∗actor_context)

  Construct a publisher object.
- Publisher (std::string name, zmq::context_t ∗actor_context, std::vector< std::string > endpoints)

  Construct a publisher object with known endpoints.
- ∼Publisher ()

  Close the publisher ZMQ socket and destroy the context.
- void bind (std::vector< std::string > new_endpoints)

  Bind the publisher to a new set of endpoints.
- std::string get_name ()

  Return the publisher name.
- void add_connection (std::string new_connection)

  Add a new endpoint to the publisher.
- void send (std::string message)

  Publish a new message.

Private Attributes

- std::string name

  Name of the publisher.
- zmq::context_t ∗ context

  ZMQ Context of the publisher.

- zmq::socket_t ∗ publisher_socket

  ZMQ Socket of the publisher.
- std::vector< std::string > endpoints

  Vector of endpoints to bind to.

6.7.1 Detailed Description

Publisher class.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 zcm::Publisher::Publisher (std::string *name,* zmq::context_t ∗ *actor_context*)

Construct a publisher object.

Parameters

in	*name*	Publisher name
in	*ZMQ*	Context of the Actor Process

6.7.2.2 zcm::Publisher::Publisher (std::string *name,* zmq::context_t ∗ *actor_context,* std::vector< std::string > *endpoints*)

Construct a publisher object with known endpoints.

Parameters

in	*name*	Publisher name
in	*ZMQ*	Context of the Actor Process
in	*endpoints*	A vector of endpoint strings

6.7.2.3 zcm::Publisher::∼Publisher ( )

Close the publisher ZMQ socket and destroy the context.

6.7.3 Member Function Documentation

6.7.3.1 void zcm::Publisher::add_connection (std::string *new_connection*)

Add a new endpoint to the publisher.

Parameters

in	*new_connection*	New endpoint to bind to

6.7.3.2 void zcm::Publisher::bind (std::vector< std::string > *new_endpoints*)

Bind the publisher to a new set of endpoints.

Parameters

in	*new_endpoints*	New set of endpoints as a vector

6.7.3.3 std::string zcm::Publisher::get_name ( )

Return the publisher name.

Returns

Publisher name

6.7.3.4 void zcm::Publisher::send (std::string *message*)

Publish a new message.

Parameters

in	*message*	The message string. Serialize complex objects to strings with protobuf

6.7.4 Member Data Documentation

6.7.4.1 zmq::context_t* zcm::Publisher::context [private]

ZMQ Context of the publisher.

6.7.4.2 std::vector<std::string> zcm::Publisher::endpoints [private]

Vector of endpoints to bind to.

6.7.4.3 std::string zcm::Publisher::name [private]

Name of the publisher.

6.7.4.4 zmq::socket_t* zcm::Publisher::publisher_socket [private]

ZMQ Socket of the publisher.

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

- /home/kelsier/GitHub/zcm/include/publisher.hpp
- /home/kelsier/GitHub/zcm/src/publisher.cpp

6.8 zcm::Server Class Reference

Server class.

`#include <server.hpp>`

Public Member Functions

- Server (std::string name, unsigned int priority, zmq::context_t ∗actor_context, std::function< void()> operation_function, Operation_Queue ∗operation_queue_ptr)

    Construct a server object.
- Server (std::string name, unsigned int priority, zmq::context_t ∗actor_context, std::vector< std::string > endpoints, std::function< void()> operation_function, Operation_Queue ∗operation_queue_ptr)

    Construct a server object with known endpoints.
- ∼Server ()

    Close the server socket and destroy the ZMQ context.
- void bind (std::vector< std::string > new_endpoints)

    Bind to a new set of endpoints param[in] new_endpoints A new vector of endpoints to bind to.
- std::string get_name ()

    Get the name of the server.
- unsigned int get_priority ()

    Get the priority of the server.
- void add_connection (std::string new_connection)

    Add a new connection to the server.
- void recv ()

    Thread function of the server Behavior: (1) Wait for a new request on the server ZMQ socket (2) Create a Server Operation (3) Enqueue onto operation_queue (4) Goto step (1)
- void rebind_operation_function (std::function< void()> new_operation_function)

    Rebind the server operation function.
- std::thread spawn ()

    Spawn a new thread for the server.
- void start ()

    Start the server thread.
- bool is_buffer_empty ()

    Is the message buffer empty?
- std::string message ()

    Is the message buffer empty?
- void set_response (std::string new_response)

    Set the response string.

Private Attributes

- std::string name

    Name of the server.
- unsigned int priority

    Priority of the server.
- zmq::context_t ∗ context

    Pointer to the server ZMQ context.
- std::vector< std::string > endpoints

    Vector of connection endpoints.
- std::function< void()> operation_function

    Operation function bound to the server - Component method that handles received requests.
- Operation_Queue ∗ operation_queue_ptr

    Pointer to the operation_queue.
- zmq::socket_t ∗ server_socket

    Pointer to the server ZMQ socket.
- bool ready

Boolean representing the state of the server to receive new requests.

- std::mutex func_mutex

  Mutex used when changing operation_function at runtime.
- std::queue< std::string > buffer

  Buffer of received messages.
- std::string ∗ response

  response string to send to client

6.8.1 Detailed Description

Server class.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 zcm::Server::Server (std::string *name*, unsigned int *priority*, zmq::context_t ∗ *actor_context*, std::function< void()> *operation_function*, Operation_Queue ∗ *operation_queue_ptr*) [inline]

Construct a server object.

Parameters

in	*name*	Server name
in	*priority*	Priority of the server
in	*ZMQ*	Context of the Actor Process
in	*operation_-function*	Operation function of the server
in	*operation_-queue_ptr*	Pointer to the operation queue

6.8.2.2 zcm::Server::Server (std::string *name*, unsigned int *priority*, zmq::context_t ∗ *actor_context*, std::vector< std::string > *endpoints*, std::function< void()> *operation_function*, Operation_Queue ∗ *operation_queue_ptr*)

Construct a server object with known endpoints.

Parameters

in	*name*	Server name
in	*priority*	Priority of the server
in	*ZMQ*	Context of the Actor Process
in	*endpoints*	A vector of endpoints to bind to
in	*operation_-function*	Operation function of the server
in	*operation_-queue_ptr*	Pointer to the operation queue

6.8.2.3 zcm::Server::∼Server ( )

Close the server socket and destroy the ZMQ context.

6.8.3 Member Function Documentation

6.8.3.1 void zcm::Server::add_connection (std::string *new_connection*)

Add a new connection to the server.

Parameters

`in`	*new_connection*	New connection address to bind to

6.8.3.2 void zcm::Server::bind (std::vector< std::string > *new_endpoints*)

Bind to a new set of endpoints param[in] new_endpoints A new vector of endpoints to bind to.

6.8.3.3 std::string zcm::Server::get_name ( )

Get the name of the server.

6.8.3.4 unsigned int zcm::Server::get_priority ( )

Get the priority of the server.

6.8.3.5 bool zcm::Server::is_buffer_empty ( )

Is the message buffer empty?

6.8.3.6 std::string zcm::Server::message ( )

Is the message buffer empty?

6.8.3.7 void zcm::Server::rebind_operation_function (std::function< void()> *new_operation_function*)

Rebind the server operation function.

Parameters

`in`	*new_operation_-function*	New server function to be handled upon recv()

6.8.3.8 void zcm::Server::recv ( )

Thread function of the server Behavior: (1) Wait for a new request on the server ZMQ socket (2) Create a Server Operation (3) Enqueue onto operation_queue (4) Goto step (1)

6.8.3.9 void zcm::Server::set_response (std::string *new_response*)

Set the response string.

6.8.3.10 std::thread zcm::Server::spawn ( )

Spawn a new thread for the server.

Returns

Server thread

6.8.3.11 void zcm::Server::start ( )

Start the server thread.

6.8.4 Member Data Documentation

6.8.4.1 std::queue<std::string> zcm::Server::buffer [private]

Buffer of received messages.

6.8.4.2 zmq::context_t* zcm::Server::context [private]

Pointer to the server ZMQ context.

6.8.4.3 std::vector<std::string> zcm::Server::endpoints [private]

Vector of connection endpoints.

6.8.4.4 std::mutex zcm::Server::func_mutex [private]

Mutex used when changing operation_function at runtime.

6.8.4.5 std::string zcm::Server::name [private]

Name of the server.

6.8.4.6 std::function<void()> zcm::Server::operation_function [private]

Operation function bound to the server - Component method that handles received requests.

6.8.4.7 Operation_Queue* zcm::Server::operation_queue_ptr [private]

Pointer to the operation_queue.

6.8.4.8 unsigned int zcm::Server::priority [private]

Priority of the server.

6.8.4.9 bool zcm::Server::ready [private]

Boolean representing the state of the server to receive new requests.

6.8.4.10 std::string* zcm::Server::response [private]

response string to send to client

6.8.4.11 zmq::socket_t∗ zcm::Server::server_socket [private]

Pointer to the server ZMQ socket.

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

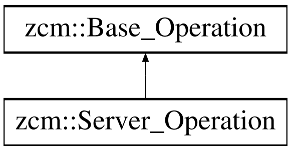
- /home/kelsier/GitHub/zcm/include/server.hpp
- /home/kelsier/GitHub/zcm/src/server.cpp

6.9 zcm::Server_Operation Class Reference

Server Operation class.

`#include <operation_types.hpp>`

Inheritance diagram for zcm::Server_Operation:



Public Member Functions

- Server_Operation (std::string name, unsigned int priority, std::function< void()> operation_function, zmq::socket_t ∗socket_ptr, bool ∗recv_ready, std::string ∗response)

  Construct a server operation.
- void execute ()

  Server operation function.
- zmq::socket_t ∗ get_socket_ptr ()

  Get the ZMQ server socket pointer.
- std::string ∗ get_response_ptr ()

  Get the response string pointer.
- void set_ready ()

  Get the ZMQ server "ready" variable.
- std::string get_name ()

  Return the operation name.
- unsigned int get_priority () const

  Return the operation priority.

Private Attributes

- std::function< void()> operation_function

  Server Operation Function.
- zmq::socket_t ∗ socket_ptr

  Pointer to the Server ZMQ socket.
- bool ∗ recv_ready

  Pointer to the Server "ready" variable.
- std::string ∗ response_ptr

  Pointer to the response string.

6.9.1 Detailed Description

Server Operation class.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 zcm::Server_Operation::Server_Operation (std::string *name*, unsigned int *priority*, std::function< void()> *operation_function*, zmq::socket_t ∗ *socket_ptr*, bool ∗ *recv_ready*, std::string ∗ *response*) `[inline]`

Construct a server operation.

Parameters

`in`	*name*	Name of the operation
`in`	*priority*	Priority of the operation
`in`	*operation_-function*	Server function
`in`	*socket_ptr*	Pointer to the Server ZMQ socket
`in`	*recv_ready*	Pointer to the Server ready variable

6.9.3 Member Function Documentation

6.9.3.1 void zcm::Server_Operation::execute ( ) `[virtual]`

Server operation function.

Reimplemented from zcm::Base_Operation.

6.9.3.2 std::string zcm::Base_Operation::get_name ( ) `[inherited]`

Return the operation name.

Returns

Name of the operation

6.9.3.3 unsigned int zcm::Base_Operation::get_priority ( ) const `[inherited]`

Return the operation priority.

Returns

Priority of the operation

6.9.3.4 std::string ∗ zcm::Server_Operation::get_response_ptr ( )

Get the response string pointer.

6.9.3.5 zmq::socket_t ∗ zcm::Server_Operation::get_socket_ptr ( )

Get the ZMQ server socket pointer.

6.9.3.6 void zcm::Server_Operation::set_ready ( )

Get the ZMQ server "ready" variable.

6.9.4 Member Data Documentation

6.9.4.1 std::function<void()> zcm::Server_Operation::operation_function [private]

Server Operation Function.

6.9.4.2 bool∗ zcm::Server_Operation::recv_ready [private]

Pointer to the Server "ready" variable.

6.9.4.3 std::string∗ zcm::Server_Operation::response_ptr [private]

Pointer to the response string.

6.9.4.4 zmq::socket_t∗ zcm::Server_Operation::socket_ptr [private]

Pointer to the Server ZMQ socket.

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

- /home/kelsier/GitHub/zcm/include/operation_types.hpp
- /home/kelsier/GitHub/zcm/src/operation_types.cpp

6.10 zcm::Subscriber Class Reference

Subscriber class.

`#include <subscriber.hpp>`

Public Member Functions

- Subscriber (std::string name, unsigned int priority, zmq::context_t ∗actor_context, std::string filter, std::function< void()> operation_function, Operation_Queue ∗operation_queue_ptr)

    Construct a subscriber object.
- Subscriber (std::string name, unsigned int priority, zmq::context_t ∗actor_context, std::string filter, std::vector< std::string > endpoints, std::function< void()> operation_function, Operation_Queue ∗operation_queue_ptr)

    Construct a subscriber object with known endpoints.
- ∼Subscriber ()

    Close the subscriber socket and destroy the ZMQ context.
- void connect (std::vector< std::string > new_endpoints)

    Connect to a new set of endpoints param[in] new_endpoints A new vector of endpoints to connect to.
- std::string get_name ()

    Get the name of the subscriber.
- unsigned int get_priority ()

    Get the priority of the subscriber.
- void add_connection (std::string new_connection)

Add a new connection to the subscriber.

- void recv ()

    Thread function of the subscriber Behavior: (1) Wait for a new message on the subscriber ZMQ socket (2) Create a Susbcriber Operation (3) Enqueue onto operation_queue (4) Goto step (1)

- void rebind_operation_function (std::function< void()> new_operation_function)

    Rebind the subscriber operation function.

- std::thread spawn ()

    Spawn a new thread for the subscriber.

- void start ()

    Start the subscriber thread.

- bool is_buffer_empty ()

    Is the message buffer empty?

- std::string message ()

    Is the message buffer empty?

Private Attributes

- std::string name

    Name of the subscriber.

- unsigned int priority

    Priority of the subscriber.

- zmq::context_t ∗ context

    Pointer to the subscriber ZMQ context.

- std::string filter

    Reception filter enforced on all received messages.

- std::vector< std::string > endpoints

    Vector of connection endpoints.

- std::function< void()> operation_function

    Operation function bound to the subscriber.

- Operation_Queue ∗ operation_queue_ptr

    Pointer to the operation queue.

- zmq::socket_t ∗ subscriber_socket

    Pointer to the subscriber ZMQ socket.

- std::mutex func_mutex

    Mutex used to change operation_function at runtime.

- std::queue< std::string > buffer

    Buffer of messages received by the subscriber.

6.10.1 Detailed Description

Subscriber class.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 zcm::Subscriber::Subscriber (std::string *name*, unsigned int *priority*, zmq::context_t ∗ *actor_context*, std::string *filter*, std::function< void()> *operation_function*, Operation_Queue ∗ *operation_queue_ptr*) [inline]

Construct a subscriber object.

Parameters

`in`	*name*	Subscriber name
`in`	*priority*	Priority of the subscriber
`in`	*ZMQ*	Context of the Actor Process
`in`	*filter*	ZMQ filter for the subscriber
`in`	*operation_-function*	Operation function of the subscriber
`in`	*operation_-queue_ptr*	Pointer to the operation queue

6.10.2.2 zcm::Subscriber::Subscriber (std::string *name*, unsigned int *priority*, zmq::context_t ∗ *actor_context*, std::string *filter*, std::vector< std::string > *endpoints*, std::function< void()> *operation_function*, Operation_Queue ∗ *operation_queue_ptr*)

Construct a subscriber object with known endpoints.

Parameters

`in`	*name*	Subscriber name
`in`	*priority*	Priority of the subscriber
`in`	*ZMQ*	Context of the Actor Process
`in`	*filter*	ZMQ filter for the subscriber
`in`	*endpoints*	A vector of endpoints to connect to
`in`	*operation_-function*	Operation function of the subscriber
`in`	*operation_-queue_ptr*	Pointer to the operation queue

6.10.2.3 zcm::Subscriber::∼Subscriber ( )

Close the subscriber socket and destroy the ZMQ context.

6.10.3 Member Function Documentation

6.10.3.1 void zcm::Subscriber::add_connection (std::string *new_connection*)

Add a new connection to the subscriber.

Parameters

`in`	*new_connection*	New connection address to connect to

6.10.3.2 void zcm::Subscriber::connect (std::vector< std::string > *new_endpoints*)

Connect to a new set of endpoints param[in] new_endpoints A new vector of endpoints to connect to.

6.10.3.3 std::string zcm::Subscriber::get_name ( )

Get the name of the subscriber.

6.10.3.4 unsigned int zcm::Subscriber::get_priority ( )

Get the priority of the subscriber.

6.10.3.5 bool zcm::Subscriber::is_buffer_empty ( )

Is the message buffer empty?

6.10.3.6 std::string zcm::Subscriber::message ( )

Is the message buffer empty?

6.10.3.7 void zcm::Subscriber::rebind_operation_function (std::function< void()> *new_operation_function*)

Rebind the subscriber operation function.

Parameters

in	*new_operation_-function*	New subscriber function to be handled upon recv()

6.10.3.8 void zcm::Subscriber::recv ( )

Thread function of the subscriber Behavior: (1) Wait for a new message on the subscriber ZMQ socket (2) Create a Susbcriber Operation (3) Enqueue onto operation_queue (4) Goto step (1)

6.10.3.9 std::thread zcm::Subscriber::spawn ( )

Spawn a new thread for the subscriber.

Returns

Subscriber thread

6.10.3.10 void zcm::Subscriber::start ( )

Start the subscriber thread.

6.10.4 Member Data Documentation

6.10.4.1 std::queue<std::string> zcm::Subscriber::buffer `[private]`

Buffer of messages received by the subscriber.

6.10.4.2 zmq::context_t* zcm::Subscriber::context `[private]`

Pointer to the subscriber ZMQ context.

6.10.4.3 std::vector<std::string> zcm::Subscriber::endpoints `[private]`

Vector of connection endpoints.

6.10.4.4 std::string zcm::Subscriber::filter `[private]`

Reception filter enforced on all received messages.

6.10.4.5 std::mutex zcm::Subscriber::func_mutex [private]

Mutex used to change operation_function at runtime.

6.10.4.6 std::string zcm::Subscriber::name [private]

Name of the subscriber.

6.10.4.7 std::function<void()> zcm::Subscriber::operation_function [private]

Operation function bound to the subscriber.

6.10.4.8 Operation_Queue* zcm::Subscriber::operation_queue_ptr [private]

Pointer to the operation queue.

6.10.4.9 unsigned int zcm::Subscriber::priority [private]

Priority of the subscriber.

6.10.4.10 zmq::socket_t* zcm::Subscriber::subscriber_socket [private]

Pointer to the subscriber ZMQ socket.

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

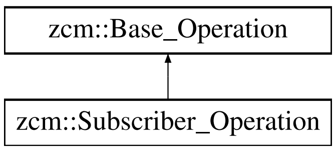
- /home/kelsier/GitHub/zcm/include/subscriber.hpp
- /home/kelsier/GitHub/zcm/src/subscriber.cpp

6.11 zcm::Subscriber_Operation Class Reference

Subscriber Operation class.

`#include <operation_types.hpp>`

Inheritance diagram for zcm::Subscriber_Operation:



Public Member Functions

- Subscriber_Operation (std::string name, unsigned int priority, std::function< void()> operation_function)

  Construct a subscriber operation.
- void execute ()

  Subscriber operation function.
- std::string get_name ()

  Return the operation name.

- unsigned int get_priority () const

  Return the operation priority.

Private Attributes

- std::function< void()> operation_function

  Subscriber Operation Function.

6.11.1 Detailed Description

Subscriber Operation class.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 zcm::Subscriber_Operation::Subscriber_Operation (std::string *name*, unsigned int *priority*, std::function< void()> *operation_function*) [inline]

Construct a subscriber operation.

Parameters

in	*name*	Name of the operation
in	*priority*	Priority of the operation
in	*operation_-function*	Subscriber function

6.11.3 Member Function Documentation

6.11.3.1 void zcm::Subscriber_Operation::execute ( ) [virtual]

Subscriber operation function.

Reimplemented from zcm::Base_Operation.

6.11.3.2 std::string zcm::Base_Operation::get_name ( ) [inherited]

Return the operation name.

Returns

Name of the operation

6.11.3.3 unsigned int zcm::Base_Operation::get_priority ( ) const [inherited]

Return the operation priority.

Returns

Priority of the operation

6.11.4 Member Data Documentation

6.11.4.1 std::function<void()> zcm::Subscriber_Operation::operation_function [private]

Subscriber Operation Function.

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

- /home/kelsier/GitHub/zcm/include/operation_types.hpp
- /home/kelsier/GitHub/zcm/src/operation_types.cpp

6.12 zcm::Timer Class Reference

Timer class.

`#include <timer.hpp>`

Public Member Functions

- Timer (std::string name, unsigned int priority, long long period, std::function< void()> operation_function, Operation_Queue *operation_queue_ptr)

  Construct a timer.
- void operation ()

  Timer thread function Behavior: (1) Wait for timer expiry (2) Create a Timer_Operation (3) Enqueue onto operation_-queue (4) Goto step (1)
- std::string get_name ()

  Get the timer name.
- unsigned int get_priority ()

  Get the timer priority.
- void change_period (long long new_period)

  Change the timer period.
- void rebind_operation_function (std::function< void()> new_operation_function)

  Rebind the timer operation function.
- std::thread spawn ()

  Spawn a new thread for the timer.
- void start ()

  Start the timer thread.

Private Attributes

- std::string name

  Name of the timer.
- unsigned int priority

  Priority of the timer.
- std::chrono::duration< long
  long, std::ratio< 1, 1000000000 > > period

  Period of the timer.
- std::function< void()> operation_function

  Operation function bound to the timer.
- Operation_Queue * operation_queue_ptr

  Pointer to the operation queue.
- std::mutex period_mutex

Mutex used to change the timer period at runtime.

- std::mutex func_mutex

  Mutex used to change the operation_function at runtime.

6.12.1 Detailed Description

Timer class.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 zcm::Timer::Timer (std::string *name*, unsigned int *priority*, long long *period*, std::function< void()> *operation_function*, Operation_Queue ∗ *operation_queue_ptr*)

Construct a timer.

Parameters

in	*name*	Name of the timer
in	*priority*	Priority of the timer
in	*period*	Period of the timer in nanoseconds
in	*operation_-function*	Operation to which the timer is bound
in	*operation_-queue_ptr*	Pointer to the operation_queue

6.12.3 Member Function Documentation

6.12.3.1 void zcm::Timer::change_period (long long *new_period*)

Change the timer period.

Parameters

in	*new_period*	New timer period in nanoseconds

6.12.3.2 std::string zcm::Timer::get_name ( )

Get the timer name.

Returns

Timer name

6.12.3.3 unsigned int zcm::Timer::get_priority ( )

Get the timer priority.

Returns

Timer priority

6.12.3.4 void zcm::Timer::operation ( )

Timer thread function Behavior: (1) Wait for timer expiry (2) Create a Timer_Operation (3) Enqueue onto operation-_queue (4) Goto step (1)

6.12.3.5 void zcm::Timer::rebind_operation_function (std::function< void()> *new_operation_function*)

Rebind the timer operation function.

Parameters

in	*new_operation_-function*	New timer function to be handled upon expiry

6.12.3.6 std::thread zcm::Timer::spawn ( )

Spawn a new thread for the timer.

Returns

Timer thread

6.12.3.7 void zcm::Timer::start ( )

Start the timer thread.

6.12.4 Member Data Documentation

6.12.4.1 std::mutex zcm::Timer::func_mutex `[private]`

Mutex used to change the operation_function at runtime.

6.12.4.2 std::string zcm::Timer::name `[private]`

Name of the timer.

6.12.4.3 std::function<void()> zcm::Timer::operation_function `[private]`

Operation function bound to the timer.

6.12.4.4 Operation_Queue∗ zcm::Timer::operation_queue_ptr `[private]`

Pointer to the operation queue.

6.12.4.5 std::chrono::duration<long long, std::ratio<1, 1000000000> > zcm::Timer::period `[private]`

Period of the timer.

6.12.4.6 std::mutex zcm::Timer::period_mutex `[private]`

Mutex used to change the timer period at runtime.

6.12.4.7 unsigned int zcm::Timer::priority `[private]`

Priority of the timer.

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

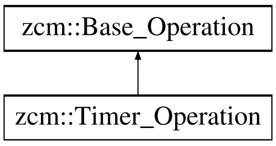
- /home/kelsier/GitHub/zcm/include/timer.hpp
- /home/kelsier/GitHub/zcm/src/timer.cpp

6.13 zcm::Timer_Operation Class Reference

Timer Operation class.

`#include <operation_types.hpp>`

Inheritance diagram for zcm::Timer_Operation:



Public Member Functions

- Timer_Operation (std::string name, unsigned int priority, std::function< void()> operation_function)

    Construct a timer operation.
- void execute ()

    Timer operation function.
- std::string get_name ()

    Return the operation name.
- unsigned int get_priority () const

    Return the operation priority.

Private Attributes

- std::function< void()> operation_function

    Timer operation function.

6.13.1 Detailed Description

Timer Operation class.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 zcm::Timer_Operation::Timer_Operation (std::string *name*, unsigned int *priority*, std::function< void()> *operation_function*) `[inline]`

Construct a timer operation.

Parameters

in	*name*	Name of the operation
in	*priority*	Priority of the operation
in	*operation_-function*	Timer function

6.13.3 Member Function Documentation

6.13.3.1 void zcm::Timer_Operation::execute ( ) [virtual]

Timer operation function.

Reimplemented from zcm::Base_Operation.

6.13.3.2 std::string zcm::Base_Operation::get_name ( ) [inherited]

Return the operation name.

Returns

Name of the operation

6.13.3.3 unsigned int zcm::Base_Operation::get_priority ( ) const [inherited]

Return the operation priority.

Returns

Priority of the operation

6.13.4 Member Data Documentation

6.13.4.1 std::function<void()> zcm::Timer_Operation::operation_function [private]

Timer operation function.

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

- /home/kelsier/GitHub/zcm/include/operation_types.hpp
- /home/kelsier/GitHub/zcm/src/operation_types.cpp

Chapter 7

File Documentation

7.1 /home/kelsier/GitHub/zcm/include/actor.hpp File Reference

This file declares the Actor class.

```
#include "json.hpp"
#include "component.hpp"
#include <dlfcn.h>
#include <fstream>
```

Classes

- class zcm::Actor

    Actor class.

Namespaces

- zcm

7.1.1 Detailed Description

This file declares the Actor class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.2 /home/kelsier/GitHub/zcm/include/client.hpp File Reference

This file declares the Client class.

```
#include <iostream>
#include <cerrno>
#include <zmq.hpp>
```

Classes

- class zcm::Client

    Client class.

Namespaces

- zcm

7.2.1 Detailed Description

This file declares the Client class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.3 /home/kelsier/GitHub/zcm/include/component.hpp File Reference

This file declares the Component class.

```
#include "timer.hpp"
#include "publisher.hpp"
#include "subscriber.hpp"
#include "client.hpp"
#include "server.hpp"
```

Classes

- class zcm::Component

    Component class.

Namespaces

- zcm

7.3.1 Detailed Description

This file declares the Component class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.4 /home/kelsier/GitHub/zcm/include/operation_queue.hpp File Reference

This file declares the Operation_Queue class.

```
#include <iostream>
#include <queue>
#include <mutex>
#include <thread>
#include <functional>
#include "operation_types.hpp"
```

Classes

- class zcm::Operation_Queue

    Operation_Queue class.
- struct zcm::Operation_Queue::PriorityOrdering

Namespaces

- zcm

7.4.1 Detailed Description

This file declares the Operation_Queue class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.5 /home/kelsier/GitHub/zcm/include/operation_types.hpp File Reference

This file declares Operation Types.

```
#include <iostream>
#include <functional>
#include "zmq.hpp"
```

Classes

- class zcm::Base_Operation

    Base Operation class.
- class zcm::Timer_Operation

    Timer Operation class.
- class zcm::Subscriber_Operation

    Subscriber Operation class.
- class zcm::Server_Operation

    Server Operation class.

Namespaces

- zcm

7.5.1 Detailed Description

This file declares Operation Types.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.6 /home/kelsier/GitHub/zcm/include/publisher.hpp File Reference

This file declares the Publisher class.

```
#include <iostream>
#include <zmq.hpp>
```

Classes

- class zcm::Publisher
  Publisher class.

Namespaces

- zcm

7.6.1 Detailed Description

This file declares the Publisher class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.7 /home/kelsier/GitHub/zcm/include/server.hpp File Reference

This file declares the Server class.

```
#include <iostream>
#include <vector>
#include <map>
#include <sstream>
#include <zmq.hpp>
#include "operation_queue.hpp"
```

Classes

- class zcm::Server

  Server class.

Namespaces

- zcm

7.7.1 Detailed Description

This file declares the Server class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.8 /home/kelsier/GitHub/zcm/include/subscriber.hpp File Reference

This file declares the Subscriber class.

```
#include <iostream>
#include <vector>
#include <map>
#include <sstream>
#include <zmq.hpp>
#include "operation_queue.hpp"
```

Classes

- class zcm::Subscriber

  Subscriber class.

Namespaces

- zcm

7.8.1 Detailed Description

This file declares the Subscriber class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.9 /home/kelsier/GitHub/zcm/include/timer.hpp File Reference

This file declares the Timer class.

```
#include <iostream>
#include <string>
#include <chrono>
#include <ratio>
#include <thread>
#include "operation_queue.hpp"
```

Classes

- class zcm::Timer
  Timer class.

Namespaces

- zcm

7.9.1 Detailed Description

This file declares the Timer class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.10 /home/kelsier/GitHub/zcm/src/actor.cpp File Reference

This file contains definitions for the Actor class.

```
#include "actor.hpp"
```

Namespaces

- zcm

7.10.1 Detailed Description

This file contains definitions for the Actor class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.11 /home/kelsier/GitHub/zcm/src/client.cpp File Reference

This file contains definitions for the Client class.

```
#include "client.hpp"
```

Namespaces

- zcm

7.11.1 Detailed Description

This file contains definitions for the Client class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.12 /home/kelsier/GitHub/zcm/src/component.cpp File Reference

This file contains definitions for the Component class.

```
#include "component.hpp"
```

Namespaces

- zcm

7.12.1 Detailed Description

This file contains definitions for the Component class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.13 /home/kelsier/GitHub/zcm/src/operation_queue.cpp File Reference

This file contains definitions for the Operation_Queue class.

```
#include "operation_queue.hpp"
```

Namespaces

- zcm

7.13.1 Detailed Description

This file contains definitions for the Operation_Queue class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.14 /home/kelsier/GitHub/zcm/src/operation_types.cpp File Reference

This file contains definitions for various Operation Types.

```
#include "operation_types.hpp"
```

Namespaces

- zcm

7.14.1 Detailed Description

This file contains definitions for various Operation Types.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.15 /home/kelsier/GitHub/zcm/src/publisher.cpp File Reference

This file contains definitions for the Publisher class.

```
#include "publisher.hpp"
```

Namespaces

- zcm

7.15.1 Detailed Description

This file contains definitions for the Publisher class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.16 /home/kelsier/GitHub/zcm/src/server.cpp File Reference

This file contains definitions for the Server class.

```
#include "server.hpp"
```

Namespaces

- zcm

7.16.1 Detailed Description

This file contains definitions for the Server class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.17 /home/kelsier/GitHub/zcm/src/subscriber.cpp File Reference

This file contains definitions for the Subscriber class.

```
#include "subscriber.hpp"
```

Namespaces

- zcm

7.17.1 Detailed Description

This file contains definitions for the Subscriber class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.18 /home/kelsier/GitHub/zcm/src/timer.cpp File Reference

This file contains definitions for the Timer class.

```
#include "timer.hpp"
```

Namespaces

- zcm

7.18.1 Detailed Description

This file contains definitions for the Timer class.

Author

Pranav Srinivas Kumar

Date

2016.04.24