

ZeroMQ Component Model

Generated by Doxygen 1.8.11

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	zcm Namespace Reference	9
6	Class Documentation	11
6.1	zcm::Base_Operation Class Reference	11
6.1.1	Detailed Description	12
6.1.2	Constructor & Destructor Documentation	12
6.1.2.1	Base_Operation(std::string name, unsigned int priority)	12
6.1.3	Member Function Documentation	12
6.1.3.1	execute()	12
6.1.3.2	get_name()	12
6.1.3.3	get_priority() const	12
6.1.4	Member Data Documentation	12
6.1.4.1	name	12

6.1.4.2	priority	13
6.2	zcm::Client Class Reference	13
6.2.1	Detailed Description	13
6.2.2	Constructor & Destructor Documentation	13
6.2.2.1	Client(std::string name)	13
6.2.2.2	Client(std::string name, std::vector< std::string > endpoints)	14
6.2.2.3	~Client()	14
6.2.3	Member Function Documentation	14
6.2.3.1	call(std::string message)	14
6.2.3.2	connect(std::vector< std::string > new_endpoints)	14
6.2.3.3	get_name()	14
6.2.4	Member Data Documentation	15
6.2.4.1	client_socket	15
6.2.4.2	context	15
6.2.4.3	endpoints	15
6.2.4.4	name	15
6.3	zcm::Component Class Reference	15
6.3.1	Detailed Description	16
6.3.2	Constructor & Destructor Documentation	16
6.3.2.1	Component()	16
6.3.2.2	~Component()	16
6.3.3	Member Function Documentation	16
6.3.3.1	spawn()	16
6.3.4	Member Data Documentation	16
6.3.4.1	executor_thread	16
6.3.4.2	operation_queue	16
6.4	zcm::Operation_Queue Class Reference	16
6.4.1	Detailed Description	17
6.4.2	Member Function Documentation	17
6.4.2.1	dequeue()	17

6.4.2.2	empty()	17
6.4.2.3	enqueue(Base_Operation *new_operation)	17
6.4.2.4	process()	17
6.4.2.5	spawn()	17
6.4.2.6	top()	17
6.4.3	Member Data Documentation	17
6.4.3.1	operation_queue	17
6.4.3.2	queue_mutex	18
6.5	zcm::Operation_Queue::PriorityOrdering Struct Reference	18
6.5.1	Member Function Documentation	18
6.5.1.1	operator()(const Base_Operation *lhs, const Base_Operation *rhs) const	18
6.6	zcm::Publisher Class Reference	18
6.6.1	Detailed Description	19
6.6.2	Constructor & Destructor Documentation	19
6.6.2.1	Publisher(std::string name)	19
6.6.2.2	Publisher(std::string name, std::vector< std::string > endpoints)	19
6.6.2.3	~Publisher()	19
6.6.3	Member Function Documentation	19
6.6.3.1	add_connection(std::string new_connection)	19
6.6.3.2	bind(std::vector< std::string > new_endpoints)	20
6.6.3.3	get_name()	20
6.6.3.4	send(std::string message)	20
6.6.4	Member Data Documentation	20
6.6.4.1	context	20
6.6.4.2	endpoints	20
6.6.4.3	name	20
6.6.4.4	publisher_socket	21
6.7	zcm::Server Class Reference	21
6.7.1	Detailed Description	22
6.7.2	Constructor & Destructor Documentation	22

6.7.2.1	Server(std::string name, unsigned int priority, std::function< std::string(const std::string &)> operation_function, Operation_Queue *operation_queue_ptr) . .	22
6.7.2.2	Server(std::string name, unsigned int priority, std::vector< std::string > endpoints, std::function< std::string(const std::string &)> operation_function, Operation_Queue *operation_queue_ptr)	22
6.7.2.3	~Server()	23
6.7.3	Member Function Documentation	23
6.7.3.1	add_connection(std::string new_connection)	23
6.7.3.2	bind(std::vector< std::string > new_endpoints)	23
6.7.3.3	get_name()	23
6.7.3.4	get_priority()	23
6.7.3.5	rebind_operation_function(std::function< std::string(const std::string &)> new← _operation_function)	23
6.7.3.6	recv()	24
6.7.3.7	spawn()	24
6.7.3.8	start()	24
6.7.4	Member Data Documentation	24
6.7.4.1	context	24
6.7.4.2	endpoints	24
6.7.4.3	func_mutex	24
6.7.4.4	name	24
6.7.4.5	operation_function	24
6.7.4.6	operation_queue_ptr	24
6.7.4.7	priority	25
6.7.4.8	ready	25
6.7.4.9	server_socket	25
6.8	zcm::Server_Operation Class Reference	25
6.8.1	Detailed Description	26
6.8.2	Constructor & Destructor Documentation	26
6.8.2.1	Server_Operation(std::string name, unsigned int priority, std::function< std← ::string(> operation_function, zmq::socket_t *socket_ptr, bool *recv_ready) . .	26
6.8.3	Member Function Documentation	26
6.8.3.1	execute()	26

6.8.3.2	get_name()	26
6.8.3.3	get_priority() const	27
6.8.3.4	get_socket_ptr()	27
6.8.3.5	set_ready()	27
6.8.4	Member Data Documentation	27
6.8.4.1	operation_function	27
6.8.4.2	recv_ready	27
6.8.4.3	socket_ptr	27
6.9	zcm::Subscriber Class Reference	27
6.9.1	Detailed Description	29
6.9.2	Constructor & Destructor Documentation	29
6.9.2.1	Subscriber(std::string name, unsigned int priority, std::string filter, std::function< void(const std::string &);> operation_function, Operation_Queue *operation_queue_ptr)	29
6.9.2.2	Subscriber(std::string name, unsigned int priority, std::string filter, std::vector< std::string > endpoints, std::function< void(const std::string &);> operation_function, Operation_Queue *operation_queue_ptr)	29
6.9.2.3	~Subscriber()	29
6.9.3	Member Function Documentation	29
6.9.3.1	add_connection(std::string new_connection)	29
6.9.3.2	connect(std::vector< std::string > new_endpoints)	30
6.9.3.3	get_name()	30
6.9.3.4	get_priority()	30
6.9.3.5	rebind_operation_function(std::function< void(const std::string &);> new_operation_function)	30
6.9.3.6	recv()	30
6.9.3.7	spawn()	30
6.9.3.8	start()	30
6.9.4	Member Data Documentation	31
6.9.4.1	context	31
6.9.4.2	endpoints	31
6.9.4.3	filter	31
6.9.4.4	func_mutex	31

6.9.4.5	name	31
6.9.4.6	operation_function	31
6.9.4.7	operation_queue_ptr	31
6.9.4.8	priority	31
6.9.4.9	subscriber_socket	31
6.10	zcm::Subscriber_Operation Class Reference	32
6.10.1	Detailed Description	32
6.10.2	Constructor & Destructor Documentation	32
6.10.2.1	Subscriber_Operation(std::string name, unsigned int priority, std::function< void()> operation_function)	32
6.10.3	Member Function Documentation	33
6.10.3.1	execute()	33
6.10.3.2	get_name()	33
6.10.3.3	get_priority() const	33
6.10.4	Member Data Documentation	33
6.10.4.1	operation_function	33
6.11	zcm::Timer Class Reference	33
6.11.1	Detailed Description	34
6.11.2	Constructor & Destructor Documentation	34
6.11.2.1	Timer(std::string name, unsigned int priority, long long period, std::function< void()> operation_function, Operation_Queue *operation_queue_ptr)	34
6.11.3	Member Function Documentation	35
6.11.3.1	change_period(long long new_period)	35
6.11.3.2	get_name()	35
6.11.3.3	get_priority()	35
6.11.3.4	operation()	35
6.11.3.5	rebind_operation_function(std::function< void()> new_operation_function)	35
6.11.3.6	spawn()	36
6.11.3.7	start()	36
6.11.4	Member Data Documentation	36
6.11.4.1	func_mutex	36

6.11.4.2	name	36
6.11.4.3	operation_function	36
6.11.4.4	operation_queue_ptr	36
6.11.4.5	period	36
6.11.4.6	period_mutex	36
6.11.4.7	priority	37
6.12	zcm::Timer_Operation Class Reference	37
6.12.1	Detailed Description	37
6.12.2	Constructor & Destructor Documentation	37
6.12.2.1	Timer_Operation(std::string name, unsigned int priority, std::function< void()> operation_function)	37
6.12.3	Member Function Documentation	38
6.12.3.1	execute()	38
6.12.3.2	get_name()	38
6.12.3.3	get_priority() const	38
6.12.4	Member Data Documentation	38
6.12.4.1	operation_function	38
7	File Documentation	39
7.1	/home/pranav/Repositories/zcm/include/client.hpp File Reference	39
7.1.1	Detailed Description	39
7.2	/home/pranav/Repositories/zcm/include/component.hpp File Reference	40
7.2.1	Detailed Description	40
7.3	/home/pranav/Repositories/zcm/include/operation_queue.hpp File Reference	40
7.3.1	Detailed Description	41
7.4	/home/pranav/Repositories/zcm/include/operation_types.hpp File Reference	41
7.4.1	Detailed Description	41
7.5	/home/pranav/Repositories/zcm/include/publisher.hpp File Reference	42
7.5.1	Detailed Description	42
7.6	/home/pranav/Repositories/zcm/include/server.hpp File Reference	42
7.6.1	Detailed Description	43

7.7	/home/pranav/Repositories/zcm/include/subscriber.hpp File Reference	43
7.7.1	Detailed Description	43
7.8	/home/pranav/Repositories/zcm/include/timer.hpp File Reference	44
7.8.1	Detailed Description	44
7.9	/home/pranav/Repositories/zcm/src/client.cpp File Reference	44
7.9.1	Detailed Description	45
7.10	/home/pranav/Repositories/zcm/src/component.cpp File Reference	45
7.10.1	Detailed Description	45
7.11	/home/pranav/Repositories/zcm/src/operation_queue.cpp File Reference	45
7.11.1	Detailed Description	46
7.12	/home/pranav/Repositories/zcm/src/operation_types.cpp File Reference	46
7.12.1	Detailed Description	46
7.13	/home/pranav/Repositories/zcm/src/publisher.cpp File Reference	46
7.13.1	Detailed Description	47
7.14	/home/pranav/Repositories/zcm/src/server.cpp File Reference	47
7.14.1	Detailed Description	47
7.15	/home/pranav/Repositories/zcm/src/subscriber.cpp File Reference	47
7.15.1	Detailed Description	48
7.16	/home/pranav/Repositories/zcm/src/timer.cpp File Reference	48
7.16.1	Detailed Description	48

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

zcm	9
-------------------------------	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

zcm::Base_Operation	11
zcm::Server_Operation	25
zcm::Subscriber_Operation	32
zcm::Timer_Operation	37
zcm::Client	13
zcm::Component	15
zcm::Operation_Queue	16
zcm::Operation_Queue::PriorityOrdering	18
zcm::Publisher	18
zcm::Server	21
zcm::Subscriber	27
zcm::Timer	33

Chapter 3

Class Index

3.1 Class List

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

zcm::Base_Operation	
Base Operation class	11
zcm::Client	
Client class	13
zcm::Component	
Component class	15
zcm::Operation_Queue	
Operation_Queue class	16
zcm::Operation_Queue::PriorityOrdering	18
zcm::Publisher	
Publisher class	18
zcm::Server	
Server class	21
zcm::Server_Operation	
Server Operation class	25
zcm::Subscriber	
Subscriber class	27
zcm::Subscriber_Operation	
Subscriber Operation class	32
zcm::Timer	
Timer class	33
zcm::Timer_Operation	
Timer Operation class	37

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/pranav/Repositories/zcm/include/ client.hpp	
This file declares the Client class	39
/home/pranav/Repositories/zcm/include/ component.hpp	
This file declares the Component class	40
/home/pranav/Repositories/zcm/include/ operation_queue.hpp	
This file declares the Operation_Queue class	40
/home/pranav/Repositories/zcm/include/ operation_types.hpp	
This file declares Operation Types	41
/home/pranav/Repositories/zcm/include/ publisher.hpp	
This file declares the Publisher class	42
/home/pranav/Repositories/zcm/include/ server.hpp	
This file declares the Server class	42
/home/pranav/Repositories/zcm/include/ subscriber.hpp	
This file declares the Subscriber class	43
/home/pranav/Repositories/zcm/include/ timer.hpp	
This file declares the Timer class	44
/home/pranav/Repositories/zcm/src/ client.cpp	
This file contains definitions for the Client class	44
/home/pranav/Repositories/zcm/src/ component.cpp	
This file contains definitions for the Component class	45
/home/pranav/Repositories/zcm/src/ operation_queue.cpp	
This file contains definitions for the Operation_Queue class	45
/home/pranav/Repositories/zcm/src/ operation_types.cpp	
This file contains definitions for various Operation Types	46
/home/pranav/Repositories/zcm/src/ publisher.cpp	
This file contains definitions for the Publisher class	46
/home/pranav/Repositories/zcm/src/ server.cpp	
This file contains definitions for the Server class	47
/home/pranav/Repositories/zcm/src/ subscriber.cpp	
This file contains definitions for the Subscriber class	47
/home/pranav/Repositories/zcm/src/ timer.cpp	
This file contains definitions for the Timer class	48

Chapter 5

Namespace Documentation

5.1 zcm Namespace Reference

Classes

- class [Base_Operation](#)
Base Operation class.
- class [Client](#)
Client class.
- class [Component](#)
Component class.
- class [Operation_Queue](#)
Operation_Queue class.
- class [Publisher](#)
Publisher class.
- class [Server](#)
Server class.
- class [Server_Operation](#)
Server Operation class.
- class [Subscriber](#)
Subscriber class.
- class [Subscriber_Operation](#)
Subscriber Operation class.
- class [Timer](#)
Timer class.
- class [Timer_Operation](#)
Timer Operation class.

Chapter 6

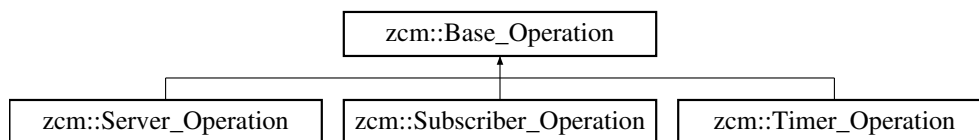
Class Documentation

6.1 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.

6.1.1 Detailed Description

Base Operation class.

6.1.2 Constructor & Destructor Documentation

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

Construct a base operation.

Parameters

in	<i>name</i>	Name of the operation
in	<i>priority</i>	Priority of the operation

6.1.3 Member Function Documentation

6.1.3.1 `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](#).

6.1.3.2 `std::string zcm::Base_Operation::get_name ()`

Return the operation name.

Returns

Name of the operation

6.1.3.3 `unsigned int zcm::Base_Operation::get_priority () const`

Return the operation priority.

Returns

Priority of the operation

6.1.4 Member Data Documentation

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

Name of the Operation.

6.1.4.2 unsigned int zcm::Base_Operation::priority [private]

Priority of the Operation.

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

- [/home/pranav/Repositories/zcm/include/operation_types.hpp](#)
- [/home/pranav/Repositories/zcm/src/operation_types.cpp](#)

6.2 zcm::Client Class Reference

[Client](#) class.

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

Public Member Functions

- [Client](#) (std::string [name](#))
Construct a client object.
- [Client](#) (std::string [name](#), std::vector< std::string > [endpoints](#))
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.
- 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.

6.2.1 Detailed Description

[Client](#) class.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 zcm::Client::Client (std::string [name](#))

Construct a client object.

Parameters

in	<i>name</i>	Client name
----	-------------	-----------------------------

6.2.2.2 `zcm::Client::Client (std::string name, std::vector< std::string > endpoints)`

Construct a client object with known endpoints.

Parameters

in	<i>name</i>	Client name
in	<i>endpoints</i>	A vector of endpoint strings

6.2.2.3 `zcm::Client::~~Client ()`

Close the client ZMQ socket and destroy the context.

6.2.3 Member Function Documentation

6.2.3.1 `std::string zcm::Client::call (std::string message)`

Call the server.

Parameters

in	<i>message</i>	The message string. Serialize complex objects to strings with protobuf
----	----------------	--

6.2.3.2 `void zcm::Client::connect (std::vector< std::string > new_endpoints)`

Connect the client to a new set of endpoints.

Parameters

in	<i>new_endpoints</i>	New set of endpoints as a vector
----	----------------------	----------------------------------

6.2.3.3 `std::string zcm::Client::get_name ()`

Return the client name.

Returns

[Client](#) name

6.2.4 Member Data Documentation

6.2.4.1 zmq::socket_t* zcm::Client::client_socket [private]

ZMQ Socket of the client.

6.2.4.2 zmq::context_t* zcm::Client::context [private]

ZMQ Context of the client.

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

Vector of endpoints to connect to.

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

Name of the publisher.

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

- /home/pranav/Repositories/zcm/include/[client.hpp](#)
- /home/pranav/Repositories/zcm/src/[client.cpp](#)

6.3 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.
- std::thread * [spawn](#) ()
Spawn the component executor thread.

Protected Attributes

- [Operation_Queue](#) * [operation_queue](#)
Pointer to the [Component](#) Operation Queue.
- std::thread * [executor_thread](#)
Pointer to the [Component](#) Executor Thread.

6.3.1 Detailed Description

[Component](#) class.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `zcm::Component::Component ()`

Construct a component Prepare the component operation queue.

6.3.2.2 `zcm::Component::~~Component ()`

Destroy the component.

6.3.3 Member Function Documentation

6.3.3.1 `std::thread * zcm::Component::spawn ()`

Spawn the component executor thread.

Returns

Return a pointer to the executor thread

6.3.4 Member Data Documentation

6.3.4.1 `std::thread* zcm::Component::executor_thread` `[protected]`

Pointer to the [Component](#) Executor Thread.

6.3.4.2 `Operation_Queue* zcm::Component::operation_queue` `[protected]`

Pointer to the [Component](#) Operation Queue.

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

- `/home/pranav/Repositories/zcm/include/component.hpp`
- `/home/pranav/Repositories/zcm/src/component.cpp`

6.4 `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.

6.4.1 Detailed Description

[Operation_Queue](#) class.

6.4.2 Member Function Documentation

6.4.2.1 void zcm::Operation_Queue::dequeue ()

6.4.2.2 bool zcm::Operation_Queue::empty ()

6.4.2.3 void zcm::Operation_Queue::enqueue ([Base_Operation](#) * new_operation)

6.4.2.4 void zcm::Operation_Queue::process ()

6.4.2.5 std::thread * zcm::Operation_Queue::spawn ()

6.4.2.6 [Base_Operation](#) * zcm::Operation_Queue::top ()

6.4.3 Member Data Documentation

6.4.3.1 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.

6.4.3.2 `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/pranav/Repositories/zcm/include/operation_queue.hpp](#)
- [/home/pranav/Repositories/zcm/src/operation_queue.cpp](#)

6.5 `zcm::Operation_Queue::PriorityOrdering` Struct Reference

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

Public Member Functions

- `bool operator()` (const [Base_Operation](#) *lhs, const [Base_Operation](#) *rhs) const

6.5.1 Member Function Documentation

6.5.1.1 `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/pranav/Repositories/zcm/include/operation_queue.hpp](#)

6.6 `zcm::Publisher` Class Reference

[Publisher](#) class.

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

Public Member Functions

- [Publisher](#) (std::string name)
Construct a publisher object.
- [Publisher](#) (std::string name, 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.

6.6.1 Detailed Description

`Publisher` class.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `zcm::Publisher::Publisher (std::string name)`

Construct a publisher object.

Parameters

in	<i>name</i>	<code>Publisher</code> name
----	-------------	-----------------------------

6.6.2.2 `zcm::Publisher::Publisher (std::string name, std::vector< std::string > endpoints)`

Construct a publisher object with known endpoints.

Parameters

in	<i>name</i>	<code>Publisher</code> name
in	<i>endpoints</i>	A vector of endpoint strings

6.6.2.3 `zcm::Publisher::~~Publisher ()`

Close the publisher ZMQ socket and destroy the context.

6.6.3 Member Function Documentation

6.6.3.1 `void zcm::Publisher::add_connection (std::string new_connection)`

Add a new endpoint to the publisher.

Parameters

in	<i>new_connection</i>	New endpoint to bind to
----	-----------------------	-------------------------

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

Bind the publisher to a new set of endpoints.

Parameters

in	<i>new_endpoints</i>	New set of endpoints as a vector
----	----------------------	----------------------------------

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

Return the publisher name.

Returns

[Publisher](#) name

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

Publish a new message.

Parameters

in	<i>message</i>	The message string. Serialize complex objects to strings with protobuf
----	----------------	--

6.6.4 Member Data Documentation

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

ZMQ Context of the publisher.

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

Vector of endpoints to bind to.

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

Name of the publisher.

6.6.4.4 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/pranav/Repositories/zcm/include/publisher.hpp
- /home/pranav/Repositories/zcm/src/publisher.cpp

6.7 zcm::Server Class Reference

[Server](#) class.

```
#include <server.hpp>
```

Public Member Functions

- [Server](#) (std::string [name](#), unsigned int [priority](#), std::function< std::string(const std::string &)> [operation_function](#), [Operation_Queue](#) *[operation_queue_ptr](#))
Construct a server object.
- [Server](#) (std::string [name](#), unsigned int [priority](#), std::vector< std::string > [endpoints](#), std::function< std::string(const std::string &)> [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< std::string(const std::string &)> [new_operation_function](#))
Rebind the server operation function.
- std::thread [spawn](#) ()
Spawn a new thread for the server.
- void [start](#) ()
Start the server thread.

Private Attributes

- `std::string name`
Name of the server.
- `unsigned int priority`
Priority of the server.
- `std::vector< std::string > endpoints`
Vector of connection endpoints.
- `std::function< std::string(const std::string &);> 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::context_t * context`
Pointer to the server ZMQ context.
- `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.

6.7.1 Detailed Description

[Server](#) class.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `zcm::Server::Server (std::string name, unsigned int priority, std::function< std::string(const std::string &);> operation_function, Operation_Queue * operation_queue_ptr) [inline]`

Construct a server object.

Parameters

in	<i>name</i>	Server name
in	<i>priority</i>	Priority of the server
in	<i>operation_function</i>	Operation function of the server
in	<i>operation_queue_ptr</i>	Pointer to the operation queue

6.7.2.2 `zcm::Server::Server (std::string name, unsigned int priority, std::vector< std::string > endpoints, std::function< std::string(const std::string &);> operation_function, Operation_Queue * operation_queue_ptr)`

Construct a server object with known endpoints.

Parameters

in	<i>name</i>	Server name
----	-------------	-----------------------------

Parameters

in	<i>priority</i>	Priority of the server
in	<i>endpoints</i>	A vector of endpoints to bind to
in	<i>operation_function</i>	Operation function of the server
in	<i>operation_queue_ptr</i>	Pointer to the operation queue

6.7.2.3 zcm::Server::~~Server ()

Close the server socket and destroy the ZMQ context.

6.7.3 Member Function Documentation

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

Add a new connection to the server.

Parameters

in	<i>new_connection</i>	New connection address to bind to
----	-----------------------	-----------------------------------

6.7.3.2 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.

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

Get the name of the server.

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

Get the priority of the server.

6.7.3.5 void zcm::Server::rebind_operation_function (std::function< std::string(const std::string &)> *new_operation_function*)

Rebind the server operation function.

Parameters

in	<i>new_operation_function</i>	New server function to be handled upon recv()
----	-------------------------------	---

6.7.3.6 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)

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

Spawn a new thread for the server.

Returns

[Server](#) thread

6.7.3.8 void zcm::Server::start ()

Start the server thread.

6.7.4 Member Data Documentation

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

Pointer to the server ZMQ context.

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

Vector of connection endpoints.

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

Mutex used when changing operation_function at runtime.

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

Name of the server.

6.7.4.5 std::function<std::string(const std::string&)> zcm::Server::operation_function [private]

Operation function bound to the server - [Component](#) method that handles received requests.

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

Pointer to the operation_queue.

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

Priority of the server.

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

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

6.7.4.9 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:

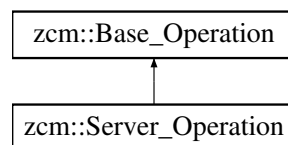
- /home/pranav/Repositories/zcm/include/server.hpp
- /home/pranav/Repositories/zcm/src/server.cpp

6.8 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< std::string()> [operation_function](#), zmq::socket_t *[socket_ptr](#), bool *[recv_ready](#))
Construct a server operation.
- void [execute](#) ()
Server operation function.
- zmq::socket_t * [get_socket_ptr](#) ()
Get the ZMQ server socket 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< std::string()>` [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.

6.8.1 Detailed Description

[Server](#) Operation class.

6.8.2 Constructor & Destructor Documentation

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

Construct a server operation.

Parameters

in	<i>name</i>	Name of the operation
in	<i>priority</i>	Priority of the operation
in	<i>operation_function</i>	Server function
in	<i>socket_ptr</i>	Pointer to the Server ZMQ socket
in	<i>recv_ready</i>	Pointer to the Server ready variable

6.8.3 Member Function Documentation

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

[Server](#) operation function.

Reimplemented from [zcm::Base_Operation](#).

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

Return the operation name.

Returns

Name of the operation

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

Return the operation priority.

Returns

Priority of the operation

6.8.3.4 `zmq::socket_t * zcm::Server_Operation::get_socket_ptr ()`

Get the ZMQ server socket pointer.

6.8.3.5 `void zcm::Server_Operation::set_ready ()`

Get the ZMQ server "ready" variable.

6.8.4 Member Data Documentation

6.8.4.1 `std::function<std::string()> zcm::Server_Operation::operation_function` `[private]`

[Server](#) Operation Function.

6.8.4.2 `bool* zcm::Server_Operation::recv_ready` `[private]`

Pointer to the [Server](#) "ready" variable.

6.8.4.3 `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/pranav/Repositories/zcm/include/operation_types.hpp](#)
- [/home/pranav/Repositories/zcm/src/operation_types.cpp](#)

6.9 zcm::Subscriber Class Reference

[Subscriber](#) class.

```
#include <subscriber.hpp>
```

Public Member Functions

- [Subscriber](#) (std::string [name](#), unsigned int [priority](#), std::string [filter](#), std::function< void(const std::string &)> [operation_function](#), [Operation_Queue](#) *[operation_queue_ptr](#))
Construct a subscriber object.
- [Subscriber](#) (std::string [name](#), unsigned int [priority](#), std::string [filter](#), std::vector< std::string > [endpoints](#), std::function< void(const std::string &)> [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 Subscriber Operation (3) Enqueue onto operation_queue (4) Goto step (1)
- void [rebind_operation_function](#) (std::function< void(const std::string &)> [new_operation_function](#))
Rebind the subscriber operation function.
- std::thread [spawn](#) ()
Spawn a new thread for the subscriber.
- void [start](#) ()
Start the subscriber thread.

Private Attributes

- std::string [name](#)
Name of the subscriber.
- unsigned int [priority](#)
Priority of the subscriber.
- std::string [filter](#)
Reception filter enforced on all received messages.
- std::vector< std::string > [endpoints](#)
Vector of connection endpoints.
- std::function< void(const std::string &)> [operation_function](#)
Operation function bound to the subscriber - [Component](#) method that handles received message.
- [Operation_Queue](#) * [operation_queue_ptr](#)
Pointer to the operation queue.
- zmq::context_t * [context](#)
Pointer to the subscriber ZMQ context.
- zmq::socket_t * [subscriber_socket](#)
Pointer to the subscriber ZMQ socket.
- std::mutex [func_mutex](#)
Mutex used to change operation_function at runtime.

6.9.1 Detailed Description

[Subscriber](#) class.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 `zcm::Subscriber::Subscriber (std::string name, unsigned int priority, std::string filter, std::function< void(const std::string &)> operation_function, Operation_Queue * operation_queue_ptr)` `[inline]`

Construct a subscriber object.

Parameters

in	<i>name</i>	Subscriber name
in	<i>priority</i>	Priority of the subscriber
in	<i>filter</i>	ZMQ filter for the subscriber
in	<i>operation_function</i>	Operation function of the subscriber
in	<i>operation_queue_ptr</i>	Pointer to the operation queue

6.9.2.2 `zcm::Subscriber::Subscriber (std::string name, unsigned int priority, std::string filter, std::vector< std::string > endpoints, std::function< void(const std::string &)> operation_function, Operation_Queue * operation_queue_ptr)`

Construct a subscriber object with known endpoints.

Parameters

in	<i>name</i>	Subscriber name
in	<i>priority</i>	Priority of the subscriber
in	<i>filter</i>	ZMQ filter for the subscriber
in	<i>endpoints</i>	A vector of endpoints to connect to
in	<i>operation_function</i>	Operation function of the subscriber
in	<i>operation_queue_ptr</i>	Pointer to the operation queue

6.9.2.3 `zcm::Subscriber::~~Subscriber ()`

Close the subscriber socket and destroy the ZMQ context.

6.9.3 Member Function Documentation

6.9.3.1 `void zcm::Subscriber::add_connection (std::string new_connection)`

Add a new connection to the subscriber.

Parameters

in	<i>new_connection</i>	New connection address to connect to
----	-----------------------	--------------------------------------

6.9.3.2 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.

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

Get the name of the subscriber.

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

Get the priority of the subscriber.

6.9.3.5 void zcm::Subscriber::rebind_operation_function (std::function< void(const std::string &);> *new_operation_function*)

Rebind the subscriber operation function.

Parameters

in	<i>new_operation_function</i>	New subscriber function to be handled upon recv()
----	-------------------------------	---

6.9.3.6 void zcm::Subscriber::recv ()

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

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

Spawn a new thread for the subscriber.

Returns

[Subscriber](#) thread

6.9.3.8 void zcm::Subscriber::start ()

Start the subscriber thread.

6.9.4 Member Data Documentation

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

Pointer to the subscriber ZMQ context.

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

Vector of connection endpoints.

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

Reception filter enforced on all received messages.

6.9.4.4 `std::mutex zcm::Subscriber::func_mutex` [private]

Mutex used to change `operation_function` at runtime.

6.9.4.5 `std::string zcm::Subscriber::name` [private]

Name of the subscriber.

6.9.4.6 `std::function<void(const std::string&)> zcm::Subscriber::operation_function` [private]

Operation function bound to the subscriber - [Component](#) method that handles received message.

6.9.4.7 `Operation_Queue* zcm::Subscriber::operation_queue_ptr` [private]

Pointer to the operation queue.

6.9.4.8 `unsigned int zcm::Subscriber::priority` [private]

Priority of the subscriber.

6.9.4.9 `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:

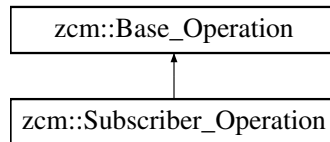
- `/home/pranav/Repositories/zcm/include/subscriber.hpp`
- `/home/pranav/Repositories/zcm/src/subscriber.cpp`

6.10 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.

6.10.1 Detailed Description

[Subscriber](#) Operation class.

6.10.2 Constructor & Destructor Documentation

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

Construct a subscriber operation.

Parameters

in	<i>name</i>	Name of the operation
in	<i>priority</i>	Priority of the operation
in	<i>operation_function</i>	Subscriber function

6.10.3 Member Function Documentation

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

[Subscriber](#) operation function.

Reimplemented from [zcm::Base_Operation](#).

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

Return the operation name.

Returns

Name of the operation

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

Return the operation priority.

Returns

Priority of the operation

6.10.4 Member Data Documentation

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

[Subscriber](#) Operation Function.

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

- [/home/pranav/Repositories/zcm/include/operation_types.hpp](#)
- [/home/pranav/Repositories/zcm/src/operation_types.cpp](#)

6.11 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.

6.11.1 Detailed Description

[Timer](#) class.

6.11.2 Constructor & Destructor Documentation

- 6.11.2.1 [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	<i>name</i>	Name of the timer
in	<i>priority</i>	Priority of the timer
in	<i>period</i>	Period of the timer in nanoseconds
in	<i>operation_function</i>	Operation to which the timer is bound
in	<i>operation_queue_ptr</i>	Pointer to the operation_queue

6.11.3 Member Function Documentation

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

Change the timer period.

Parameters

in	<i>new_period</i>	New timer period in nanoseconds
----	-------------------	---------------------------------

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

Get the timer name.

Returns

[Timer](#) name

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

Get the timer priority.

Returns

[Timer](#) priority

6.11.3.4 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)

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

Rebind the timer operation function.

Parameters

in	<code>new_operation_function</code>	New timer function to be handled upon expiry
----	-------------------------------------	--

6.11.3.6 `std::thread zcm::Timer::spawn ()`

Spawn a new thread for the timer.

Returns

[Timer](#) thread

6.11.3.7 `void zcm::Timer::start ()`

Start the timer thread.

6.11.4 Member Data Documentation

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

Mutex used to change the operation_function at runtime.

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

Name of the timer.

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

Operation function bound to the timer.

6.11.4.4 `Operation_Queue* zcm::Timer::operation_queue_ptr` [private]

Pointer to the operation queue.

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

Period of the timer.

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

Mutex used to change the timer period at runtime.

6.11.4.7 unsigned int zcm::Timer::priority [private]

Priority of the timer.

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

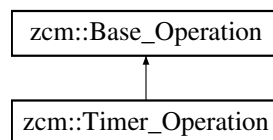
- /home/pranav/Repositories/zcm/include/timer.hpp
- /home/pranav/Repositories/zcm/src/timer.cpp

6.12 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.

6.12.1 Detailed Description

Timer Operation class.

6.12.2 Constructor & Destructor Documentation

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

Construct a timer operation.

Parameters

in	<i>name</i>	Name of the operation
in	<i>priority</i>	Priority of the operation
in	<i>operation_function</i>	Timer function

6.12.3 Member Function Documentation

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

[Timer](#) operation function.

Reimplemented from [zcm::Base_Operation](#).

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

Return the operation name.

Returns

Name of the operation

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

Return the operation priority.

Returns

Priority of the operation

6.12.4 Member Data Documentation

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

[Timer](#) operation function.

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

- [/home/pranav/Repositories/zcm/include/operation_types.hpp](#)
- [/home/pranav/Repositories/zcm/src/operation_types.cpp](#)

Chapter 7

File Documentation

7.1 /home/pranav/Repositories/zcm/include/client.hpp File Reference

This file declares the Client class.

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

Classes

- class `zcm::Client`
Client class.

Namespaces

- `zcm`

7.1.1 Detailed Description

This file declares the Client class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.2 /home/pranav/Repositories/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](#)

7.2.1 Detailed Description

This file declares the Component class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.3 /home/pranav/Repositories/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](#)

7.3.1 Detailed Description

This file declares the `Operation_Queue` class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.4 /home/pranav/Repositories/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](#)

7.4.1 Detailed Description

This file declares Operation Types.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.5 /home/pranav/Repositories/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](#)

7.5.1 Detailed Description

This file declares the Publisher class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.6 /home/pranav/Repositories/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](#)

7.6.1 Detailed Description

This file declares the Server class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.7 /home/pranav/Repositories/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](#)

7.7.1 Detailed Description

This file declares the Subscriber class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.8 /home/pranav/Repositories/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](#)

7.8.1 Detailed Description

This file declares the Timer class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.9 /home/pranav/Repositories/zcm/src/client.cpp File Reference

This file contains definitions for the Client class.

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

Namespaces

- [zcm](#)

7.9.1 Detailed Description

This file contains definitions for the Client class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.10 /home/pranav/Repositories/zcm/src/component.cpp File Reference

This file contains definitions for the Component class.

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

Namespaces

- [zcm](#)

7.10.1 Detailed Description

This file contains definitions for the Component class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.11 /home/pranav/Repositories/zcm/src/operation_queue.cpp File Reference

This file contains definitions for the Operation_Queue class.

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

Namespaces

- [zcm](#)

7.11.1 Detailed Description

This file contains definitions for the `Operation_Queue` class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.12 `/home/pranav/Repositories/zcm/src/operation_types.cpp` File Reference

This file contains definitions for various Operation Types.

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

Namespaces

- [zcm](#)

7.12.1 Detailed Description

This file contains definitions for various Operation Types.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.13 `/home/pranav/Repositories/zcm/src/publisher.cpp` File Reference

This file contains definitions for the `Publisher` class.

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

Namespaces

- [zcm](#)

7.13.1 Detailed Description

This file contains definitions for the Publisher class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.14 /home/pranav/Repositories/zcm/src/server.cpp File Reference

This file contains definitions for the Server class.

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

Namespaces

- [zcm](#)

7.14.1 Detailed Description

This file contains definitions for the Server class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.15 /home/pranav/Repositories/zcm/src/subscriber.cpp File Reference

This file contains definitions for the Subscriber class.

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

Namespaces

- [zcm](#)

7.15.1 Detailed Description

This file contains definitions for the Subscriber class.

Author

Pranav Srinivas Kumar

Date

2016.04.24

7.16 /home/pranav/Repositories/zcm/src/timer.cpp File Reference

This file contains definitions for the Timer class.

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

Namespaces

- [zcm](#)

7.16.1 Detailed Description

This file contains definitions for the Timer class.

Author

Pranav Srinivas Kumar

Date

2016.04.24