

## ZeroMQ Component Model

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	zcm Namespace Reference . . . . .	9
<b>6</b>	<b>Class Documentation</b>	<b>11</b>
6.1	zcm::Actor Class Reference . . . . .	11
6.1.1	Detailed Description . . . . .	11
6.1.2	Member Function Documentation . . . . .	11
6.1.2.1	configure(std::string configuration_file) . . . . .	11
6.1.2.2	get_name() . . . . .	12
6.1.2.3	run() . . . . .	12
6.1.3	Member Data Documentation . . . . .	12
6.1.3.1	component_instances . . . . .	12
6.1.3.2	name . . . . .	12
6.2	zcm::Base_Operation Class Reference . . . . .	12

6.2.1	Detailed Description	13
6.2.2	Constructor & Destructor Documentation	13
6.2.2.1	Base_Operation(std::string name, unsigned int priority)	13
6.2.3	Member Function Documentation	13
6.2.3.1	execute()	13
6.2.3.2	get_name()	13
6.2.3.3	get_priority() const	14
6.2.4	Member Data Documentation	14
6.2.4.1	name	14
6.2.4.2	priority	14
6.3	zcm::Client Class Reference	14
6.3.1	Detailed Description	15
6.3.2	Constructor & Destructor Documentation	15
6.3.2.1	Client(std::string name)	15
6.3.2.2	Client(std::string name, std::vector< std::string > endpoints)	15
6.3.2.3	~Client()	15
6.3.3	Member Function Documentation	15
6.3.3.1	call(std::string message)	15
6.3.3.2	connect(std::vector< std::string > new_endpoints)	16
6.3.3.3	get_name()	16
6.3.4	Member Data Documentation	16
6.3.4.1	client_socket	16
6.3.4.2	context	16
6.3.4.3	endpoints	16
6.3.4.4	name	16
6.4	zcm::Component Class Reference	17
6.4.1	Detailed Description	18
6.4.2	Constructor & Destructor Documentation	18
6.4.2.1	Component()	18
6.4.2.2	~Component()	18

6.4.3	Member Function Documentation . . . . .	18
6.4.3.1	add_client(Client *new_client) . . . . .	18
6.4.3.2	add_publisher(Publisher *new_publisher) . . . . .	19
6.4.3.3	add_server(Server *new_server) . . . . .	19
6.4.3.4	add_subscriber(Subscriber *new_subscriber) . . . . .	19
6.4.3.5	add_timer(Timer *new_timer) . . . . .	19
6.4.3.6	client(std::string client_name) . . . . .	19
6.4.3.7	configure_clients(std::map< std::string, std::vector< std::string >> client_endpoints) . . . . .	20
6.4.3.8	configure_publishers(std::map< std::string, std::vector< std::string >> publisher_endpoints) . . . . .	20
6.4.3.9	configure_servers(std::map< std::string, std::vector< std::string >> server_endpoints) . . . . .	20
6.4.3.10	configure_subscribers(std::map< std::string, std::vector< std::string >> subscriber_endpoints) . . . . .	20
6.4.3.11	get_operation_queue() . . . . .	20
6.4.3.12	publisher(std::string publisher_name) . . . . .	21
6.4.3.13	register_server_operation(std::string operation_name, std::function< std::string(const std::string &);> operation_function) . . . . .	21
6.4.3.14	register_subscriber_operation(std::string operation_name, std::function< void(const std::string &);> operation_function) . . . . .	21
6.4.3.15	register_timer_operation(std::string operation_name, std::function< void();> operation_function) . . . . .	21
6.4.3.16	server(std::string server_name) . . . . .	22
6.4.3.17	spawn() . . . . .	22
6.4.3.18	subscriber(std::string subscriber_name) . . . . .	22
6.4.3.19	timer(std::string timer_name) . . . . .	22
6.4.4	Member Data Documentation . . . . .	22
6.4.4.1	clients . . . . .	22
6.4.4.2	executor_thread . . . . .	22
6.4.4.3	operation_queue . . . . .	23
6.4.4.4	publishers . . . . .	23
6.4.4.5	server_functions . . . . .	23

6.4.4.6	servers	23
6.4.4.7	subscriber_functions	23
6.4.4.8	subscribers	23
6.4.4.9	timer_functions	23
6.4.4.10	timers	23
6.5	zcm::Operation_Queue Class Reference	23
6.5.1	Detailed Description	24
6.5.2	Member Function Documentation	24
6.5.2.1	dequeue()	24
6.5.2.2	empty()	24
6.5.2.3	enqueue(Base_Operation *new_operation)	24
6.5.2.4	process()	24
6.5.2.5	spawn()	24
6.5.2.6	top()	24
6.5.3	Member Data Documentation	24
6.5.3.1	operation_queue	24
6.5.3.2	queue_mutex	25
6.6	zcm::Operation_Queue::PriorityOrdering Struct Reference	25
6.6.1	Member Function Documentation	25
6.6.1.1	operator()(const Base_Operation *lhs, const Base_Operation *rhs) const	25
6.7	zcm::Publisher Class Reference	25
6.7.1	Detailed Description	26
6.7.2	Constructor & Destructor Documentation	26
6.7.2.1	Publisher(std::string name)	26
6.7.2.2	Publisher(std::string name, std::vector< std::string > endpoints)	26
6.7.2.3	~Publisher()	26
6.7.3	Member Function Documentation	26
6.7.3.1	add_connection(std::string new_connection)	26
6.7.3.2	bind(std::vector< std::string > new_endpoints)	27
6.7.3.3	get_name()	27

6.7.3.4	<a href="#">send(std::string message)</a>	27
6.7.4	<a href="#">Member Data Documentation</a>	27
6.7.4.1	<a href="#">context</a>	27
6.7.4.2	<a href="#">endpoints</a>	27
6.7.4.3	<a href="#">name</a>	27
6.7.4.4	<a href="#">publisher_socket</a>	28
6.8	<a href="#">zcm::Server Class Reference</a>	28
6.8.1	<a href="#">Detailed Description</a>	29
6.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	29
6.8.2.1	<a href="#">Server(std::string name, unsigned int priority, std::function&lt; std::string(const std::string &amp;)&gt; operation_function, Operation_Queue *operation_queue_ptr)</a>	29
6.8.2.2	<a href="#">Server(std::string name, unsigned int priority, std::vector&lt; std::string &gt; endpoints, std::function&lt; std::string(const std::string &amp;)&gt; operation_function, Operation_Queue *operation_queue_ptr)</a>	29
6.8.2.3	<a href="#">~Server()</a>	30
6.8.3	<a href="#">Member Function Documentation</a>	30
6.8.3.1	<a href="#">add_connection(std::string new_connection)</a>	30
6.8.3.2	<a href="#">bind(std::vector&lt; std::string &gt; new_endpoints)</a>	30
6.8.3.3	<a href="#">get_name()</a>	30
6.8.3.4	<a href="#">get_priority()</a>	30
6.8.3.5	<a href="#">rebind_operation_function(std::function&lt; std::string(const std::string &amp;)&gt; new_operation_function)</a>	30
6.8.3.6	<a href="#">recv()</a>	31
6.8.3.7	<a href="#">spawn()</a>	31
6.8.3.8	<a href="#">start()</a>	31
6.8.4	<a href="#">Member Data Documentation</a>	31
6.8.4.1	<a href="#">context</a>	31
6.8.4.2	<a href="#">endpoints</a>	31
6.8.4.3	<a href="#">func_mutex</a>	31
6.8.4.4	<a href="#">name</a>	31
6.8.4.5	<a href="#">operation_function</a>	31
6.8.4.6	<a href="#">operation_queue_ptr</a>	31

6.8.4.7	priority	32
6.8.4.8	ready	32
6.8.4.9	server_socket	32
6.9	zcm::Server_Operation Class Reference	32
6.9.1	Detailed Description	33
6.9.2	Constructor & Destructor Documentation	33
6.9.2.1	Server_Operation(std::string name, unsigned int priority, std::function< std::string()> operation_function, zmq::socket_t *socket_ptr, bool *recv_ready)	33
6.9.3	Member Function Documentation	33
6.9.3.1	execute()	33
6.9.3.2	get_name()	33
6.9.3.3	get_priority() const	34
6.9.3.4	get_socket_ptr()	34
6.9.3.5	set_ready()	34
6.9.4	Member Data Documentation	34
6.9.4.1	operation_function	34
6.9.4.2	recv_ready	34
6.9.4.3	socket_ptr	34
6.10	zcm::Subscriber Class Reference	34
6.10.1	Detailed Description	36
6.10.2	Constructor & Destructor Documentation	36
6.10.2.1	Subscriber(std::string name, unsigned int priority, std::string filter, std::function< void(const std::string &);> operation_function, Operation_Queue *operation_queue_ptr)	36
6.10.2.2	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)	36
6.10.2.3	~Subscriber()	36
6.10.3	Member Function Documentation	36
6.10.3.1	add_connection(std::string new_connection)	36
6.10.3.2	connect(std::vector< std::string > new_endpoints)	37
6.10.3.3	get_name()	37
6.10.3.4	get_priority()	37



6.10.3.5	<code>rebind_operation_function(std::function&lt; void(const std::string &amp;)&gt; new_↵ operation_function)</code>	37
6.10.3.6	<code>recv()</code>	37
6.10.3.7	<code>spawn()</code>	37
6.10.3.8	<code>start()</code>	37
6.10.4	Member Data Documentation	38
6.10.4.1	<code>context</code>	38
6.10.4.2	<code>endpoints</code>	38
6.10.4.3	<code>filter</code>	38
6.10.4.4	<code>func_mutex</code>	38
6.10.4.5	<code>name</code>	38
6.10.4.6	<code>operation_function</code>	38
6.10.4.7	<code>operation_queue_ptr</code>	38
6.10.4.8	<code>priority</code>	38
6.10.4.9	<code>subscriber_socket</code>	38
6.11	<code>zcm::Subscriber_Operation</code> Class Reference	39
6.11.1	Detailed Description	39
6.11.2	Constructor & Destructor Documentation	39
6.11.2.1	<code>Subscriber_Operation(std::string name, unsigned int priority, std::function&lt; void(&gt; operation_function)</code>	39
6.11.3	Member Function Documentation	40
6.11.3.1	<code>execute()</code>	40
6.11.3.2	<code>get_name()</code>	40
6.11.3.3	<code>get_priority() const</code>	40
6.11.4	Member Data Documentation	40
6.11.4.1	<code>operation_function</code>	40
6.12	<code>zcm::Timer</code> Class Reference	40
6.12.1	Detailed Description	41
6.12.2	Constructor & Destructor Documentation	41
6.12.2.1	<code>Timer(std::string name, unsigned int priority, long long period, std::function&lt; void(&gt; operation_function, Operation_Queue *operation_queue_ptr)</code>	41
6.12.3	Member Function Documentation	42

6.12.3.1	<code>change_period(long long new_period)</code>	42
6.12.3.2	<code>get_name()</code>	42
6.12.3.3	<code>get_priority()</code>	42
6.12.3.4	<code>operation()</code>	42
6.12.3.5	<code>rebind_operation_function(std::function&lt; void()&gt; new_operation_function)</code>	42
6.12.3.6	<code>spawn()</code>	43
6.12.3.7	<code>start()</code>	43
6.12.4	Member Data Documentation	43
6.12.4.1	<code>func_mutex</code>	43
6.12.4.2	<code>name</code>	43
6.12.4.3	<code>operation_function</code>	43
6.12.4.4	<code>operation_queue_ptr</code>	43
6.12.4.5	<code>period</code>	43
6.12.4.6	<code>period_mutex</code>	43
6.12.4.7	<code>priority</code>	44
6.13	<code>zcm::Timer_Operation</code> Class Reference	44
6.13.1	Detailed Description	44
6.13.2	Constructor & Destructor Documentation	44
6.13.2.1	<code>Timer_Operation(std::string name, unsigned int priority, std::function&lt; void()&gt; operation_function)</code>	44
6.13.3	Member Function Documentation	45
6.13.3.1	<code>execute()</code>	45
6.13.3.2	<code>get_name()</code>	45
6.13.3.3	<code>get_priority() const</code>	45
6.13.4	Member Data Documentation	45
6.13.4.1	<code>operation_function</code>	45

<b>7 File Documentation</b>	<b>47</b>
7.1 /home/pranav/Repositories/zcm/include/actor.hpp File Reference	47
7.1.1 Detailed Description	47
7.2 /home/pranav/Repositories/zcm/include/client.hpp File Reference	48
7.2.1 Detailed Description	48
7.3 /home/pranav/Repositories/zcm/include/component.hpp File Reference	48
7.3.1 Detailed Description	49
7.4 /home/pranav/Repositories/zcm/include/operation_queue.hpp File Reference	49
7.4.1 Detailed Description	49
7.5 /home/pranav/Repositories/zcm/include/operation_types.hpp File Reference	50
7.5.1 Detailed Description	50
7.6 /home/pranav/Repositories/zcm/include/publisher.hpp File Reference	50
7.6.1 Detailed Description	51
7.7 /home/pranav/Repositories/zcm/include/server.hpp File Reference	51
7.7.1 Detailed Description	51
7.8 /home/pranav/Repositories/zcm/include/subscriber.hpp File Reference	52
7.8.1 Detailed Description	52
7.9 /home/pranav/Repositories/zcm/include/timer.hpp File Reference	52
7.9.1 Detailed Description	53
7.10 /home/pranav/Repositories/zcm/src/actor.cpp File Reference	53
7.10.1 Detailed Description	53
7.11 /home/pranav/Repositories/zcm/src/client.cpp File Reference	53
7.11.1 Detailed Description	54
7.12 /home/pranav/Repositories/zcm/src/component.cpp File Reference	54
7.12.1 Detailed Description	54
7.13 /home/pranav/Repositories/zcm/src/operation_queue.cpp File Reference	54
7.13.1 Detailed Description	55
7.14 /home/pranav/Repositories/zcm/src/operation_types.cpp File Reference	55
7.14.1 Detailed Description	55
7.15 /home/pranav/Repositories/zcm/src/publisher.cpp File Reference	55
7.15.1 Detailed Description	56
7.16 /home/pranav/Repositories/zcm/src/server.cpp File Reference	56
7.16.1 Detailed Description	56
7.17 /home/pranav/Repositories/zcm/src/subscriber.cpp File Reference	56
7.17.1 Detailed Description	57
7.18 /home/pranav/Repositories/zcm/src/timer.cpp File Reference	57
7.18.1 Detailed Description	57



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">zcm</a> . . . . .	9
-------------------------------	---



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

zcm::Actor . . . . .	11
zcm::Base_Operation . . . . .	12
zcm::Server_Operation . . . . .	32
zcm::Subscriber_Operation . . . . .	39
zcm::Timer_Operation . . . . .	44
zcm::Client . . . . .	14
zcm::Component . . . . .	17
zcm::Operation_Queue . . . . .	23
zcm::Operation_Queue::PriorityOrdering . . . . .	25
zcm::Publisher . . . . .	25
zcm::Server . . . . .	28
zcm::Subscriber . . . . .	34
zcm::Timer . . . . .	40





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">zcm::Actor</a>	
<a href="#">Actor</a> class . . . . .	11
<a href="#">zcm::Base_Operation</a>	
Base Operation class . . . . .	12
<a href="#">zcm::Client</a>	
Client class . . . . .	14
<a href="#">zcm::Component</a>	
Component class . . . . .	17
<a href="#">zcm::Operation_Queue</a>	
Operation_Queue class . . . . .	23
<a href="#">zcm::Operation_Queue::PriorityOrdering</a>	25
<a href="#">zcm::Publisher</a>	
Publisher class . . . . .	25
<a href="#">zcm::Server</a>	
Server class . . . . .	28
<a href="#">zcm::Server_Operation</a>	
Server Operation class . . . . .	32
<a href="#">zcm::Subscriber</a>	
Subscriber class . . . . .	34
<a href="#">zcm::Subscriber_Operation</a>	
Subscriber Operation class . . . . .	39
<a href="#">zcm::Timer</a>	
Timer class . . . . .	40
<a href="#">zcm::Timer_Operation</a>	
Timer Operation class . . . . .	44



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/pranav/Repositories/zcm/include/ <a href="#">actor.hpp</a>	47
This file declares the Actor class	
/home/pranav/Repositories/zcm/include/ <a href="#">client.hpp</a>	48
This file declares the Client class	
/home/pranav/Repositories/zcm/include/ <a href="#">component.hpp</a>	48
This file declares the Component class	
/home/pranav/Repositories/zcm/include/ <a href="#">operation_queue.hpp</a>	49
This file declares the Operation_Queue class	
/home/pranav/Repositories/zcm/include/ <a href="#">operation_types.hpp</a>	50
This file declares Operation Types	
/home/pranav/Repositories/zcm/include/ <a href="#">publisher.hpp</a>	50
This file declares the Publisher class	
/home/pranav/Repositories/zcm/include/ <a href="#">server.hpp</a>	51
This file declares the Server class	
/home/pranav/Repositories/zcm/include/ <a href="#">subscriber.hpp</a>	52
This file declares the Subscriber class	
/home/pranav/Repositories/zcm/include/ <a href="#">timer.hpp</a>	52
This file declares the Timer class	
/home/pranav/Repositories/zcm/src/ <a href="#">actor.cpp</a>	53
This file contains definitions for the Actor class	
/home/pranav/Repositories/zcm/src/ <a href="#">client.cpp</a>	53
This file contains definitions for the Client class	
/home/pranav/Repositories/zcm/src/ <a href="#">component.cpp</a>	54
This file contains definitions for the Component class	
/home/pranav/Repositories/zcm/src/ <a href="#">operation_queue.cpp</a>	54
This file contains definitions for the Operation_Queue class	
/home/pranav/Repositories/zcm/src/ <a href="#">operation_types.cpp</a>	55
This file contains definitions for various Operation Types	
/home/pranav/Repositories/zcm/src/ <a href="#">publisher.cpp</a>	55
This file contains definitions for the Publisher class	
/home/pranav/Repositories/zcm/src/ <a href="#">server.cpp</a>	56
This file contains definitions for the Server class	
/home/pranav/Repositories/zcm/src/ <a href="#">subscriber.cpp</a>	56
This file contains definitions for the Subscriber class	
/home/pranav/Repositories/zcm/src/ <a href="#">timer.cpp</a>	57
This file contains definitions for the Timer class	



## Chapter 5

# Namespace Documentation

### 5.1 zcm Namespace Reference

#### Classes

- class [Actor](#)  
*Actor class.*
- 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::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	<i>configuration_file</i>	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:

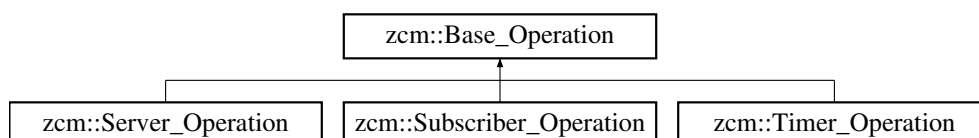
- [/home/pranav/Repositories/zcm/include/actor.hpp](#)
- [/home/pranav/Repositories/zcm/src/actor.cpp](#)

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	<i>name</i>	Name of the operation
in	<i>priority</i>	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/pranav/Repositories/zcm/include/operation\\_types.hpp](#)
- [/home/pranav/Repositories/zcm/src/operation\\_types.cpp](#)

## 6.3 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.*

### 6.3.1 Detailed Description

[Client](#) class.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 `zcm::Client::Client ( std::string name )`

Construct a client object.

##### Parameters

in	<i>name</i>	<a href="#">Client</a> name
----	-------------	-----------------------------

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

Construct a client object with known endpoints.

##### Parameters

in	<i>name</i>	<a href="#">Client</a> name
in	<i>endpoints</i>	A vector of endpoint strings

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

**6.3.3.3** std::string zcm::Client::get\_name ( )

Return the client name.

## Returns

[Client](#) name

## 6.3.4 Member Data Documentation

**6.3.4.1** zmq::socket\_t\* zcm::Client::client\_socket [private]

ZMQ Socket of the client.

**6.3.4.2** zmq::context\_t\* zcm::Client::context [private]

ZMQ Context of the client.

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

Vector of endpoints to connect to.

**6.3.4.4** 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](#)

## 6.4 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\\_timer\\_operation](#) (std::string operation\_name, std::function< void()> operation\_function)  
*Register a timer operation.*
- void [register\\_subscriber\\_operation](#) (std::string operation\_name, std::function< void(const std::string &)> operation\_function)  
*Register a subscriber operation.*
- void [register\\_server\\_operation](#) (std::string operation\_name, std::function< std::string(const std::string &)> operation\_function)  
*Register a server operation.*
- std::thread \* [spawn](#) ()  
*Spawn the component executor thread.*

## Public Attributes

- `std::map< std::string, std::function< void()> > timer_functions`  
*A map of timer operations.*
- `std::map< std::string, std::function< void(const std::string &);> > subscriber_functions`  
*A map of subscriber operations.*
- `std::map< std::string, std::function< std::string(const std::string &);> > server_functions`  
*A map of server 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	<i>new_client</i>	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	<i>new_publisher</i>	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	<i>new_server</i>	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	<i>new_subscriber</i>	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	<i>new_timer</i>	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	<i>client_name</i>	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	<i>client_endpoints</i>	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	<i>publisher_endpoints</i>	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	<i>server_endpoints</i>	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	<i>subscriber_endpoints</i>	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	<i>publisher_name</i>	Name of the publisher
----	-----------------------	-----------------------

**6.4.3.13 void zcm::Component::register\_server\_operation ( std::string *operation\_name*, std::function< std::string(const std::string &)> *operation\_function* )**

Register a server operation.

**Parameters**

in	<i>operation_name</i>	Name of the server operation
in	<i>operation_function</i>	The actual server operation function

**6.4.3.14 void zcm::Component::register\_subscriber\_operation ( std::string *operation\_name*, std::function< void(const std::string &)> *operation\_function* )**

Register a subscriber operation.

**Parameters**

in	<i>operation_name</i>	Name of the subscriber operation
in	<i>operation_function</i>	The actual subscriber operation function

**6.4.3.15 void zcm::Component::register\_timer\_operation ( std::string *operation\_name*, std::function< void()> *operation\_function* )**

Register a timer operation.

**Parameters**

in	<i>operation_name</i>	Name of the timer operation
in	<i>operation_function</i>	The actual timer operation function

**6.4.3.16 Server \* zcm::Component::server ( std::string *server\_name* )**

Get a component server by name.

**Parameters**

in	<i>server_name</i>	Name of the server
----	--------------------	--------------------

**6.4.3.17 std::thread \* zcm::Component::spawn ( )**

Spawn the component executor thread.

**Returns**

Return a pointer to the executor thread

**6.4.3.18 Subscriber \* zcm::Component::subscriber ( std::string *subscriber\_name* )**

Get a component subscriber by name.

**Parameters**

in	<i>subscriber_name</i>	Name of the subscriber
----	------------------------	------------------------

**6.4.3.19 Timer \* zcm::Component::timer ( std::string *timer\_name* )**

Get a component timer by name.

**Parameters**

in	<i>timer_name</i>	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** `Operation_Queue* zcm::Component::operation_queue` [protected]

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

**6.4.4.4** `std::vector<Publisher*> zcm::Component::publishers` [protected]

A vector of component publishers.

**6.4.4.5** `std::map<std::string, std::function<std::string(const std::string&)>> > zcm::Component::server_functions`

A map of server operations.

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

A vector of component servers.

**6.4.4.7** `std::map<std::string, std::function<void(const std::string&)>> > zcm::Component::subscriber_functions`

A map of subscriber operations.

**6.4.4.8** `std::vector<Subscriber*> zcm::Component::subscribers` [protected]

A vector of component subscribers.

**6.4.4.9** `std::map<std::string, std::function<void()>> > zcm::Component::timer_functions`

A map of timer operations.

**6.4.4.10** `std::vector<Timer*> zcm::Component::timers` [protected]

A vector of component timers.

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`

## 6.5 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/pranav/Repositories/zcm/include/operation\_queue.hpp
- /home/pranav/Repositories/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/pranav/Repositories/zcm/include/operation\_queue.hpp

## 6.7 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.*

### 6.7.1 Detailed Description

`Publisher` class.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 `zcm::Publisher::Publisher ( std::string name )`

Construct a publisher object.

##### Parameters

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

#### 6.7.2.2 `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

#### 6.7.2.3 `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	<i>new_connection</i>	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	<i>new_endpoints</i>	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	<i>message</i>	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/pranav/Repositories/zcm/include/publisher.hpp`
- `/home/pranav/Repositories/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`, 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.*

### 6.8.1 Detailed Description

[Server](#) class.

### 6.8.2 Constructor & Destructor Documentation

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

Construct a server object.

#### Parameters

in	<i>name</i>	<a href="#">Server</a> 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

**6.8.2.2** `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>	<a href="#">Server</a> 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

6.8.2.3 `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	<i>new_connection</i>	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 `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 <a href="#">recv()</a>
----	-------------------------------	---

#### 6.8.3.6 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)

#### 6.8.3.7 std::thread zcm::Server::spawn ( )

Spawn a new thread for the server.

##### Returns

[Server](#) thread

#### 6.8.3.8 void zcm::Server::start ( )

Start the server thread.

### 6.8.4 Member Data Documentation

#### 6.8.4.1 zmq::context\_t\* zcm::Server::context [private]

Pointer to the server ZMQ context.

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

Vector of connection endpoints.

#### 6.8.4.3 std::mutex zcm::Server::func\_mutex [private]

Mutex used when changing operation\_function at runtime.

#### 6.8.4.4 std::string zcm::Server::name [private]

Name of the server.

#### 6.8.4.5 std::function<std::string(const std::string&)> zcm::Server::operation\_function [private]

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

#### 6.8.4.6 Operation\_Queue\* zcm::Server::operation\_queue\_ptr [private]

Pointer to the operation\_queue.

#### 6.8.4.7 unsigned int zcm::Server::priority [private]

Priority of the server.

#### 6.8.4.8 bool zcm::Server::ready [private]

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

#### 6.8.4.9 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:

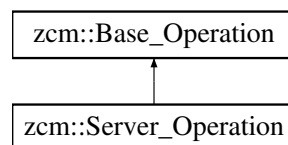
- /home/pranav/Repositories/zcm/include/[server.hpp](#)
- /home/pranav/Repositories/zcm/src/[server.cpp](#)

## 6.9 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< 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.*

### 6.9.1 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< 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>	<a href="#">Server</a> function
in	<i>socket_ptr</i>	Pointer to the <a href="#">Server</a> ZMQ socket
in	<i>recv_ready</i>	Pointer to the <a href="#">Server</a> 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** `zmq::socket_t * zcm::Server_Operation::get_socket_ptr ( )`

Get the ZMQ server socket pointer.

**6.9.3.5** `void zcm::Server_Operation::set_ready ( )`

Get the ZMQ server "ready" variable.

## 6.9.4 Member Data Documentation

**6.9.4.1** `std::function<std::string()> 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** `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](#)

## 6.10 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.*

### 6.10.1 Detailed Description

[Subscriber](#) class.

### 6.10.2 Constructor & Destructor Documentation

6.10.2.1 `zcm::Subscriber::Subscriber ( std::string name, unsigned int priority, 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>	<a href="#">Subscriber</a> 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

6.10.2.2 `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>	<a href="#">Subscriber</a> 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

6.10.2.3 `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	<i>new_connection</i>	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 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 <a href="#">recv()</a>
----	-------------------------------	---

6.10.3.6 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)

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

Spawn a new thread for the subscriber.

## Returns

[Subscriber](#) thread

6.10.3.8 void zcm::Subscriber::start ( )

Start the subscriber thread.

## 6.10.4 Member Data Documentation

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

Pointer to the subscriber ZMQ context.

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

Vector of connection endpoints.

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

Reception filter enforced on all received messages.

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

Mutex used to change `operation_function` at runtime.

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

Name of the subscriber.

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

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

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

Pointer to the operation queue.

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

Priority of the subscriber.

6.10.4.9 `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:

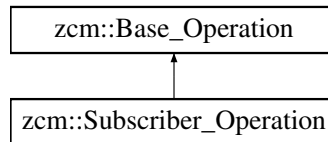
- [/home/pranav/Repositories/zcm/include/subscriber.hpp](#)
- [/home/pranav/Repositories/zcm/src/subscriber.cpp](#)

## 6.11 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	<i>name</i>	Name of the operation
in	<i>priority</i>	Priority of the operation
in	<i>operation_function</i>	<a href="#">Subscriber</a> 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/pranav/Repositories/zcm/include/operation\\_types.hpp](#)
- [/home/pranav/Repositories/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	<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

## 6.12.3 Member Function Documentation

6.12.3.1 void zcm::Timer::change\_period ( long long *new\_period* )

Change the timer period.

## Parameters

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

## 6.12.3.2 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	<i>new_operation_function</i>	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:

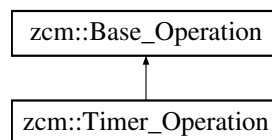
- /home/pranav/Repositories/zcm/include/timer.hpp
- /home/pranav/Repositories/zcm/src/timer.cpp

## 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	<i>name</i>	Name of the operation
in	<i>priority</i>	Priority of the operation
in	<i>operation_function</i>	<a href="#">Timer</a> 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&lt;void()&gt; 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/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/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](#)

#### 7.2.1 Detailed Description

This file declares the Client class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.3 /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](#)

### 7.3.1 Detailed Description

This file declares the Component class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.4 /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](#)

### 7.4.1 Detailed Description

This file declares the Operation\_Queue class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.5 /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](#)

### 7.5.1 Detailed Description

This file declares Operation Types.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.6 /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](#)

### 7.6.1 Detailed Description

This file declares the Publisher class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.7 /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](#)

### 7.7.1 Detailed Description

This file declares the Server class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.8 /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](#)

#### 7.8.1 Detailed Description

This file declares the Subscriber class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.9 /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](#)

### 7.9.1 Detailed Description

This file declares the Timer class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.10 /home/pranav/Repositories/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/pranav/Repositories/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/pranav/Repositories/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/pranav/Repositories/zcm/src/operation\_queue.cpp File Reference

This file contains definitions for the Operation\_Queue class.

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

## Namespaces

- [zcm](#)

### 7.13.1 Detailed Description

This file contains definitions for the Operation\_Queue class.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.14 /home/pranav/Repositories/zcm/src/operation\_types.cpp File Reference

This file contains definitions for various Operation Types.

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

## Namespaces

- [zcm](#)

### 7.14.1 Detailed Description

This file contains definitions for various Operation Types.

#### Author

Pranav Srinivas Kumar

#### Date

2016.04.24

## 7.15 /home/pranav/Repositories/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/pranav/Repositories/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/pranav/Repositories/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/pranav/Repositories/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

